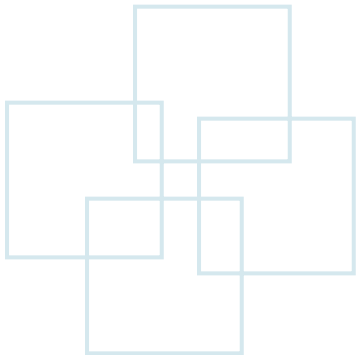


Chapter 13

File Handling

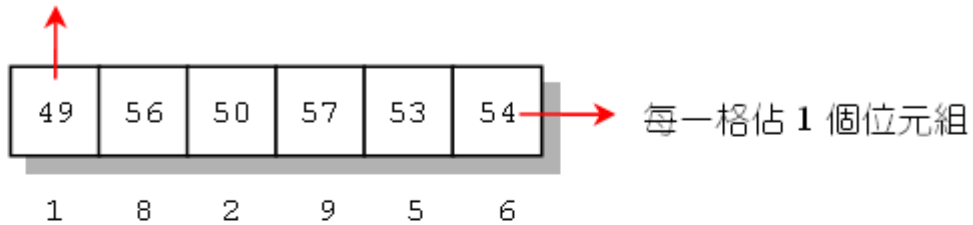




File Types

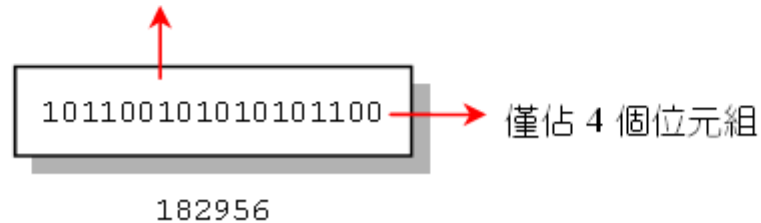
將數值 182956
以文字檔儲存

以 ASCII 碼儲存



將數值 182956 以二
進位的格式儲存

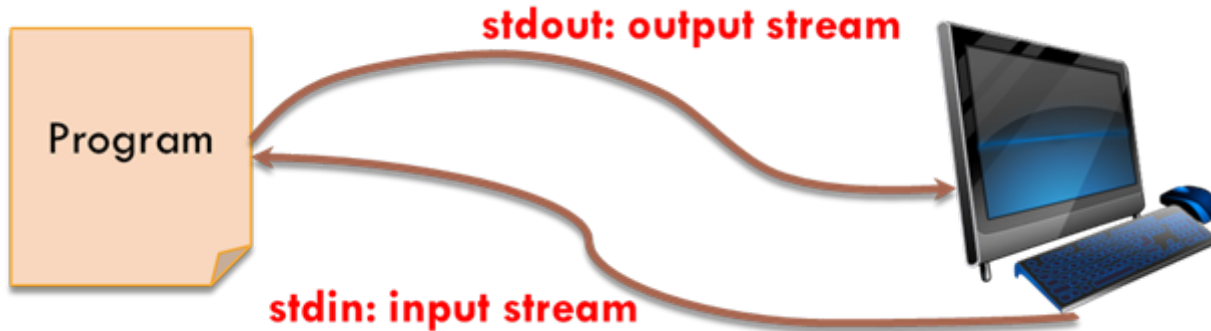
以二進位碼儲存



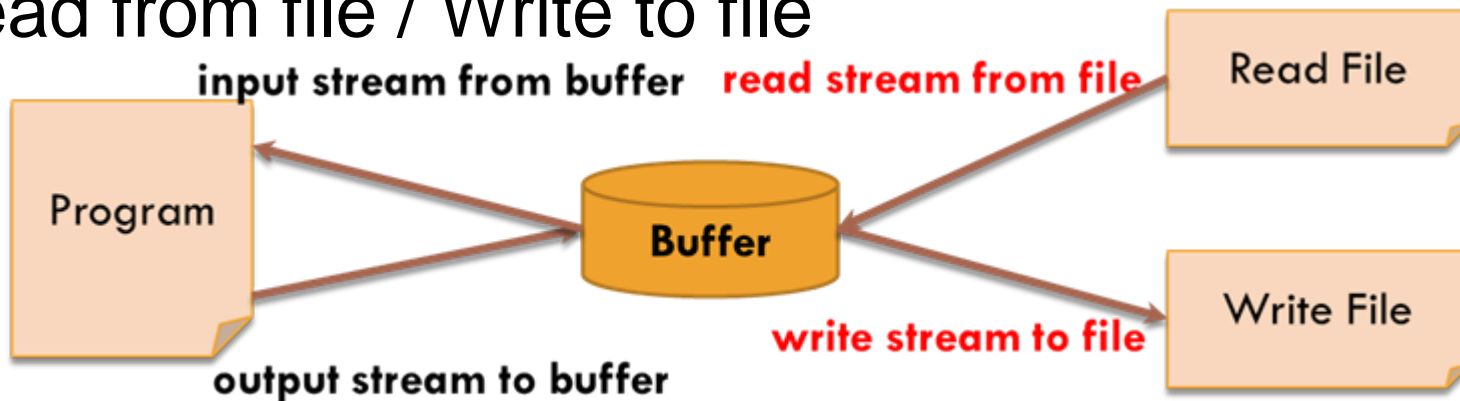


Files and Streams

- Read from stdin / Write to stdout



- Read from file / Write to file

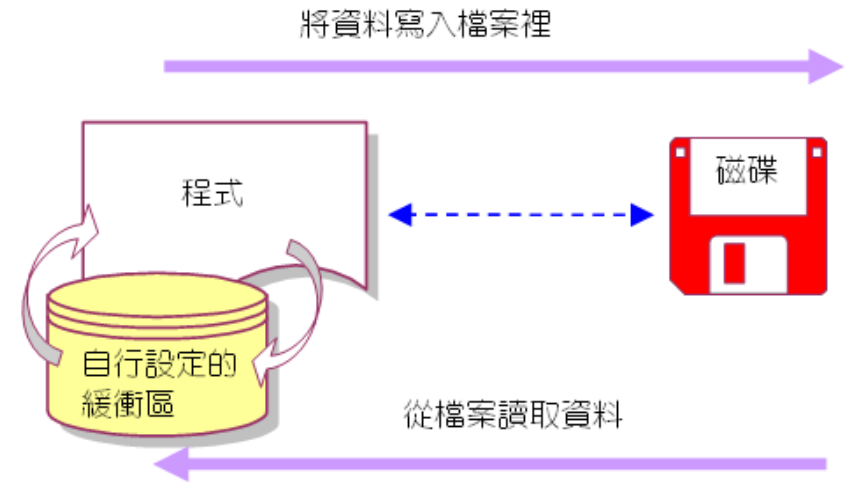
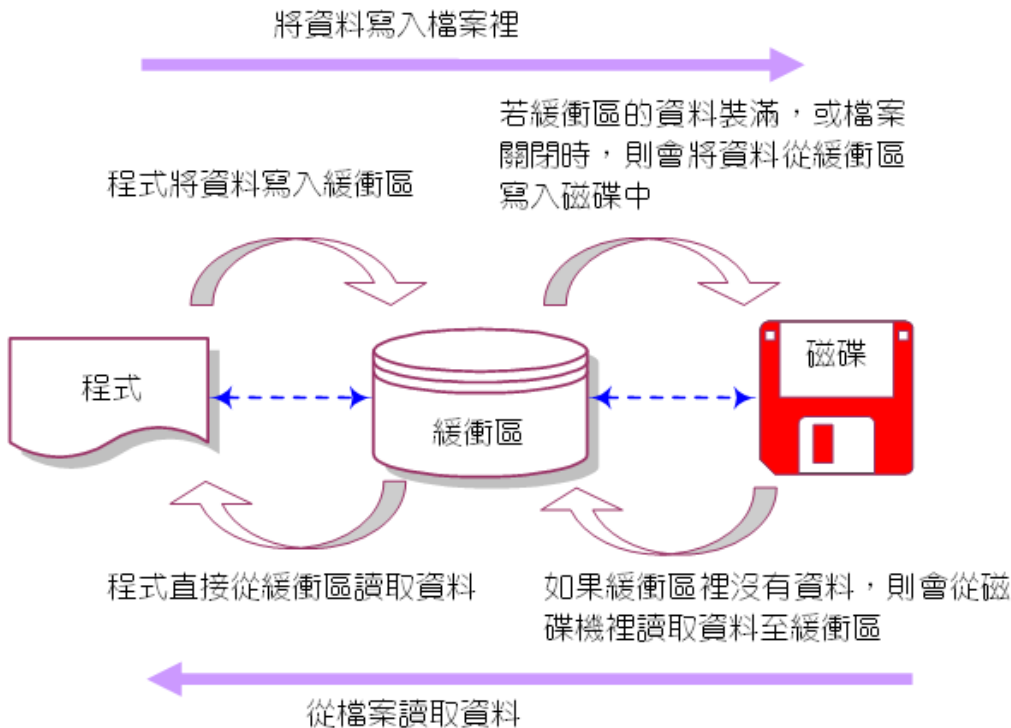


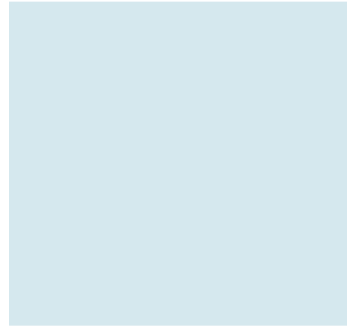
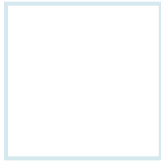


Buffered vs. Non-buffered File Handling

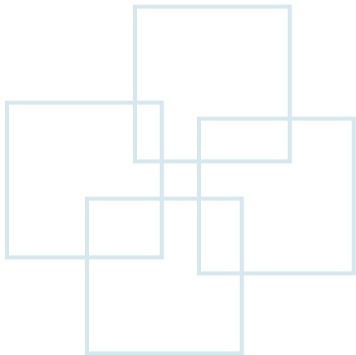
- Buffered:
有緩衝區的檔案處理：

- Nonbuffered:
無緩衝區的檔案處理：





Buffered File Handling





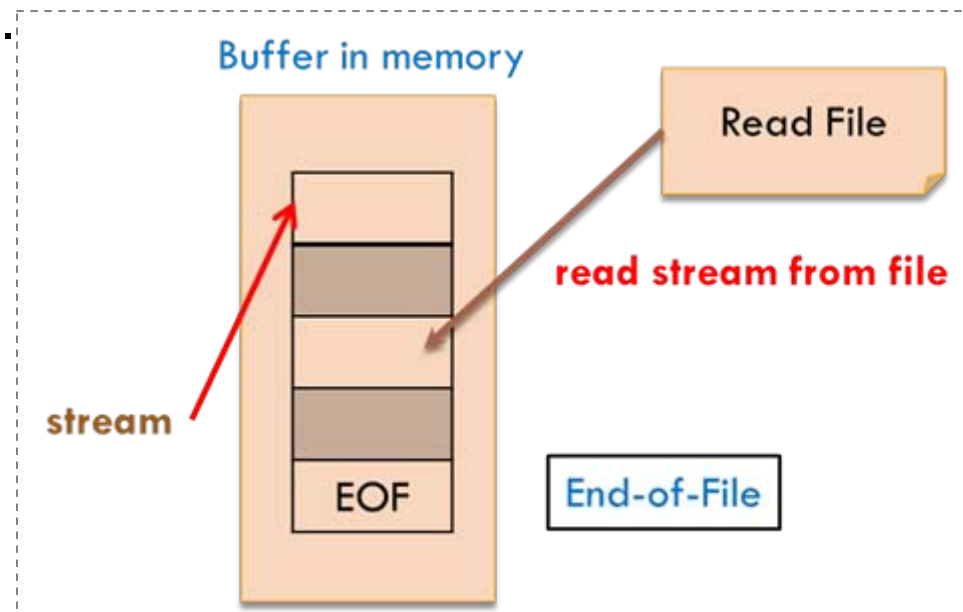
Open/Close Files with Buffers

• Open Files

- `FILE *stream;`
`stream = fopen("file name", "open mode");`
 - `fopen()` returns a FILE pointer to the beginning of the file data.
 - If open fails, NULL returned.

• Close files

- `fclose(stream);`





File Opening Modes

- "r": Open a file for reading. The file must exist.
- "w": Create an empty file for writing. If a file with the same name already exists its content is erased and the file is treated as a new empty file.
- "a": Append to a file. Writing operations append data at the end of the file. The file is created if it does not exist.
- When open a file as a *binary file*, a "b" character has to be included in the *mode string*.
 - "rb", "wb", "ab"

存取模式	代碼	說明
讀取資料	r	開啟檔案以供讀取。在開啟前，此檔案必須先存在於磁碟機內。如果檔案不存在，則開檔函數 <code>fopen()</code> 開檔失敗，將無法執行
寫入資料	w	開啟檔案以供寫入。如果檔案已經存在，則該檔案的內容將被覆蓋掉。如果檔案不存在，則系統會自行建立此檔案
附加於檔案之後	a	開啟一個檔案，可將資料寫入此檔案的末端。如果檔案不存在，則系統會自行建立此檔案



Open Files with Buffer: Example

```
#include <stdio.h>
int main() {
    /* open a file for writing called "test.txt" */
    FILE *stream = fopen("test.txt", "w");
    fclose(stream); // close file
    return 0;
}
```

Default folder is the same with where the program is executed

\ is the escape symbol

File name can be an
absolute path: "D:\\dir\\test.txt"
relative path: "test.txt"



Open Files with Buffers: Example (Cont.)

```

FILE *fptr;                /* 宣告指向檔案的指標 fptr */
fptr=fopen("abc.txt", "r"); /* 開啟檔案 abc.txt 以供讀取 */
if (fptr != NULL)          /* 判別檔案是否開啟成功 */
{
    /* 檔案開啟成功時，所要執行的程式碼 */
}
else
{
    /* 檔案開啟失敗時，所要執行的程式碼 */
}

```



Handling Functions to Files with Buffer

- #include <stdio.h>

函數功能	格式及說明
開啟檔案	<pre>FILE *fopen(const char *filename, const char *mode);</pre> 開啟指定的檔案，並指定存取模式。 <code>fopen()</code> 的第一個引數為檔案名稱字串，第二個引數為存取模式的代表。 <code>fopen()</code> 的傳回值為檔案指標，開檔失敗傳回 <code>NULL</code>
關閉檔案	<pre>int fclose(FILE *fptr);</pre> 關閉由 <code>fptr</code> 所指向的檔案，關檔成功傳回 <code>0</code>
讀取字元	<pre>int getc(FILE *fptr);</pre> 由 <code>fptr</code> 所指向的檔案讀取一個字元，傳回值為被讀取的字元
寫入字元	<pre>int putc(int ch, FILE *fptr);</pre> 將字元 <code>ch</code> 寫入由 <code>fptr</code> 所指向的檔案



Handling Functions to Files with Buffer (Cont.)

函數功能	格式及說明
讀取字串	<pre>char *fgets(char *str,int maxchar,FILE *fptr);</pre> <p>從 fptr 所指向的檔案裡讀取最多 maxchar 個字元，然後將它寫入字元陣列 str 中。若讀取失敗，或已讀到檔尾，則傳回 NULL</p>
寫入字串	<pre>int fputs(const char *str,FILE *fptr);</pre> <p>將字串 str 寫入 fptr 所指向的檔案</p>
檢查檔案是否結束	<pre>int feof(FILE *fptr);</pre> <p>檢查 fptr 所指向的檔案是否已讀取到檔案結束的位置。若尚未到達檔尾，則傳回 0；若已到檔尾，則傳回非 0 的值</p>
區塊輸入	<pre>size_t fread(void *p,size_t s,size_t cnt,FILE *fptr);</pre> <p>由檔案讀取 cnt 個資料項目，存放到指標 p 所指向的位址中，每一個資料項目的大小為 s 個位元組，傳回值為讀取資料的個數</p>
區塊輸出	<pre>size_t fwrite(const void *p,size_t s,size_t cnt,FILE *fptr);</pre> <p>將 cnt 個大小為 s 個位元組的資料，寫入指標 p 所指向的位址中，傳回值為成功寫入資料的個數</p>

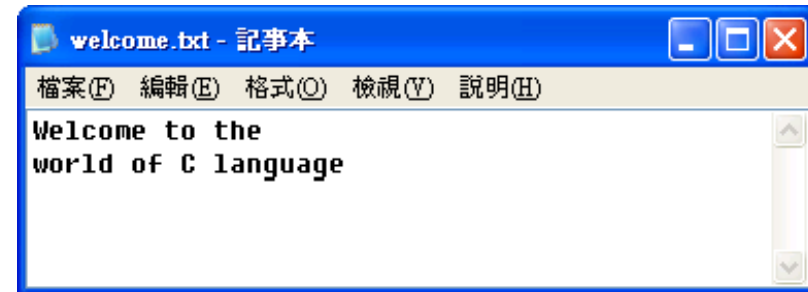


Example: Reading File

```

01  /* 顯示檔案內容，並計算字元數 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06  FILE *fptr;          /* 宣告指向檔案的指標 fptr */
07  char ch;            /* 宣告字元變數 ch，用來接收讀取的字元 */
08  int count=0;       /* 宣告整數 count，用來計算檔案的字元數 */
09
10  fptr=fopen("c:\\prog\\welcome.txt", "r");    /* 開啟檔案 */
11  if(fptr!=NULL) /* 如果 fopen() 的傳回值不為 NULL，代表檔案開啟成功 */
12  {
13      while((ch=getc(fptr))!=EOF) /* 判斷是否到達檔尾 */
14      {
15          printf("%c", ch);      /* 一次印出一個字元 */
16          count++;
17      }
18      fclose(fptr);             /* 關閉所開啟的檔案 */
19      printf("\n總共有%d個字元\n", count);
20  }
21  else /* 檔案開啟失敗 */
22      printf("檔案開啟失敗!!\n");
23
24  system("pause");
25  return 0;
26  }

```



/* OUTPUT---

Welcome to the
world of C language
總共有 34 個字元

*/

空白與換行字元也列入字元數的計算，因此總字元數為 34：

W	e	l	c	o	m	e		t	o		t	h	e	\n
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

w	o	r	l	d		o	f		C		l	a	n	g	u	a	g	e	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	ang

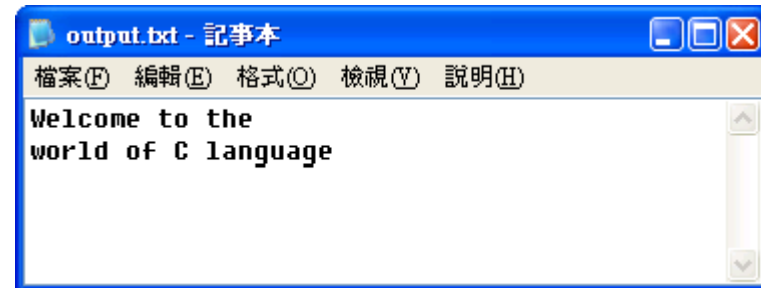


Example: Copying Files

```

01  /* 拷貝檔案內容到其它的檔案 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      FILE *fptr1,*fptr2;      /* 宣告指向檔案的指標 fptr1 與 fptr2 */
07      char ch;
08      fptr1=fopen("c:\\prog\\welcome.txt","r"); /* 開啟可讀取的檔案 */
09      fptr2=fopen("c:\\prog\\output.txt","w"); /* 開啟可寫入的檔案 */
10      if((fptr1!=NULL) && (fptr2!=NULL))      /* 如果開檔成功 */
11      {
12          while((ch=getc(fptr1))!=EOF)        /* 判斷是否到達檔尾 */
13              putchar(ch,fptr2);            /* 將字元 ch 寫到 fptr2 所指向的檔案 */
14          fclose(fptr1);                      /* 關閉 fptr1 所指向的檔案 */
15          fclose(fptr2);                      /* 關閉 fptr2 所指向的檔案 */
16          printf("檔案拷貝完成!!\n");
17      }
18      else
19          printf("檔案開啟失敗!!\n");
20      system("pause");
21      return 0;
22  }

```



/* OUTPUT--

檔案拷貝完成!!

***/**



Example: Appending Files

```

01  /* 由鍵盤輸入字串，並附加到檔案 output.txt 中 */
02  #include <stdio.h>
03  #include <conio.h>
04  #include <stdlib.h>
05  #define ENTER 13
06  #define MAX 80
07  int main(void)
08  {
09      FILE *fptr;
10      char str[MAX],ch; /* 宣告字元陣列 str，用來儲存由鍵盤輸入的字串 */
11      int i=0;
12      fptr=fopen("c:\\prog\\output.txt","a");
13      printf("請輸入字串，按 ENTER 鍵結束輸入:\n");
14      while((ch=getche())!=ENTER && i<MAX) /* 按下的鍵不是 ENTER 且 i<MAX */
15          str[i++]=ch; /* 一次增加一個字元到字元陣列 str 中 */
16      putc('\n',fptr); /* 寫入換行字元 */
17      fwrite(str,sizeof(char),i,fptr);
18      fclose(fptr);
19      printf("\n 檔案附加完成!!\n");
20      system("pause");
21      return 0;
22  }

```

```

/* OUTPUT-----
請輸入字串，按 ENTER 鍵結束輸入：
I love the C language best!
檔案附加完成!!
-----*/

```



Reading Files with fread()

```

01  /* 使用 fread() 函數讀取檔案內容 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define MAX 80
05  int main(void)
06  {
07      FILE *fptr;
08      char str[MAX];
09      int bytes;          /* 存放 fread() 成功讀取的字元數 */
10      fptr=fopen("c:\\prog\\output.txt","r");
11      while(!feof(fptr)) /* 如果還沒讀到檔尾 */
12      {
13          bytes=fread(str,sizeof(char),MAX-1 ,fptr);
14
15          str[bytes]='\0';
16          printf("%s\n",str); /* 印出檔案內容 */
17      }
18      fclose(fptr); /* 關閉檔案 */
19      system("pause");
20      return 0;
21  }

```

```

/* OUTPUT-----
Welcome to the
world of C language
I love the C language best!
-----*/

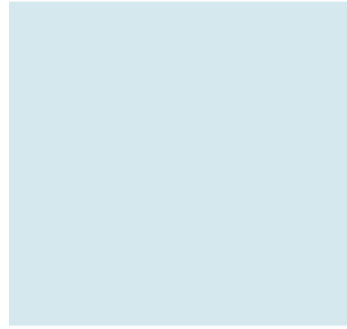
```



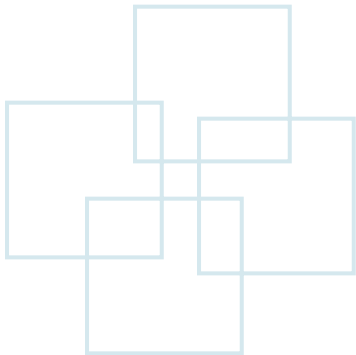

fscanf() and fprintf()

- Use fscanf() to read from file:
 - Useful for skipping spaces ('\n', '\t', ' ')
 - Read and ignore any spaces (including blank spaces and the newline and tab characters) encountered before the next spaces
 - On success, the function returns the number of items successfully read
- Use fprintf() to write to file

```
FILE *in, *out;  
in = fopen("input.txt", "r");  
out = fopen("output.txt", "w");  
char str[100];  
int cnt;  
while ( !feof(in) ) {  
    cnt = fscanf (in, "%s", str); // read from file  
    if ( cnt > 0 ) fprintf(out, "%s\n", str); // write to file  
}
```

Non-buffered File Handling





Opening Files

open("FileName", Mode, Attribute);

open("檔案名稱", 開啟模式, 存取屬性);

檔案開啟模式		說 明
基本 模 式	O_RDONLY	開啟的檔案只供讀取，不能寫入資料
	O_WRONLY	開啟的檔案只供寫入，不能讀取資料
	O_RDWR	開啟的檔案可供讀取與寫入資料
修 飾 模 式	O_CREAT	若開啟的檔案不存在，則建立新檔；若存在，則此功能無效
	O_APPEND	開啟的檔案可供寫入，寫入時不會蓋掉原有的內容，而是附加在其後，若與 O_RDONLY 一起使用，則此功能無效
	O_BINARY	開啟一個二進位檔案 (binary file)
	O_TEXT	開啟文字檔案



Modes and Attributes

• Use | to concatenate modes

O_WRONLY /* 開啟寫檔，此檔只供寫入，不能讀取 */

O_WRONLY | O_APPEND /* 開啟寫檔，此檔可以附加資料，但不能讀取 */

O_WRONLY | O_CREAT | O_APPEND /* 若檔案不存在，則建立可附加資料的新檔 */

O_RDONLY | O_TEXT /* 開啟已存在的文字檔，且只供讀取 */

open("FileName", **O_CREAT**, **Attribute**);

open("檔案名稱", **O_CREAT**, **存取屬性**);

存取屬性	說明
S_IWRITE	新建立的檔案可供寫入
S_IREAD	新建立的檔案只供讀取（即屬性為唯讀）
S_IREAD S_IWRITE	新建立的檔案，可供讀取與寫入資料



Handling Functions

函數功能	格式
開啟檔案	<code>int open(const char *filename, int oflag[, int pmode]);</code> 開啟指定的檔案及開啟模式，傳回值為檔案代號，開檔失敗時傳回-1
關閉檔案	<code>int close(int handle);</code> 關閉指定的檔案，關檔成功傳回 0，關檔失敗傳回 1
開新檔案	<code>int creat(const char *filename, int pmode);</code> 建立一個存取屬性為 pmode 的檔案，傳回值為檔案代號，開檔失敗時傳回-1
讀取資料	<code>int read(int handle, char *buffer, unsigned count);</code> 讀取資料，最多一次讀取 count 位元組，並存放到位址為 buffer 的變數裡。傳回值為實際讀取資料的位元組，若是傳回-1，表示讀取失敗
寫入資料	<code>int write(int handle, char *buffer, unsigned count);</code> 將位址為 buffer 的變數內容寫入檔案中，最多可一次寫入 count 位元組，傳回值為實際寫入資料的位元組，若是傳回-1，表示寫入失敗



Example: Copying File

```
01  /* 複製檔案內容 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #include <fcntl.h>
05  #include <io.h>
06  #include <sys/stat.h>
07  #define SIZE 512          /* 設定 read() 一次可讀取的最大位元組為 512 */
08  int main(void)
09  {
10      char buffer[SIZE];
11      int f1,f2,bytes;
12
13      f1=open("c:\\prog\\welcome.txt",O_RDONLY|O_TEXT);
14      f2=creat("c:\\prog\\output2.txt",S_IWRITE);
15
```

Libraries providing handling functions





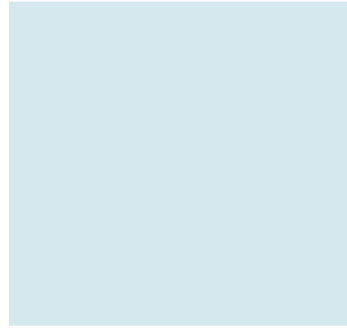
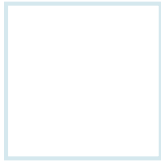
Example: Copying File (Cont.)

```
16     if((f1!=-1)&&(f2!=-1))        /* 測試檔案是否開啟成功 */
17     {
18         while(!eof(f1))          /* 如果還沒有讀到檔案末端 */
19         {
20             bytes=read(f1,buffer,SIZE);    /* 從 f1 讀取資料 */
21             write(f2,buffer,bytes);        /* 將資料寫入檔案 f1 中 */
22         }
23         close(f1);
24         close(f2);
25         printf("檔案拷貝完成!!\n");
26     }
27     else
28         printf("檔案開啟失敗!!\n");
29
30     system("pause");
31     return 0;
32 }
```

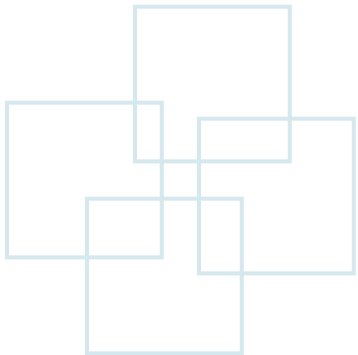
/* **OUTPUT**---

檔案拷貝完成!!

-----*/



Handling Binary Files





Using fopen() to Handle Binary Files

- Using buffered file handling with *fopen()*:

存取模式	代碼	說明
二進位檔的讀取	rb	開啟一個僅供讀取資料的二進位檔案 (binary file)
二進位檔的寫入	wb	開啟一個僅供寫入資料的二進位檔案
二進位檔的附加	ab	開啟一個可以附加資料的二進位檔案

```

/* 開啟可供附加資料的二進位檔案test.bin */
FILE *fptr;
fptr=fopen("test.bin", "ab");

```




Writing Data to Binary Files (Buffered)

```

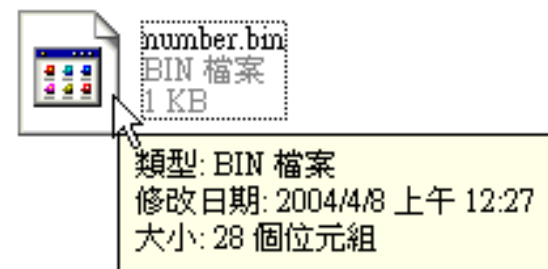
01  /* 輸入資料到二進位檔案 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      double a=3.14,b=6.28;
07      int arr[]={12,43,64};
08      FILE *fptr;
09
10      fptr=fopen("c:\\prog\\number.bin","wb");      /* 開啟檔案 */
11      fwrite(&a,sizeof(double),1,fptr);          /* 寫入變數 a 的值 */
12      fwrite(&b,sizeof(double),1,fptr);          /* 寫入變數 b 的值 */
13      fwrite(arr,sizeof(int),3,fptr);            /* 寫入陣列 arr 的所有元素 */
14      fclose(fptr);      /* 關閉檔案 */
15      printf("檔案寫入完成!!\n");
16      system("pause");
17      return 0;
18  }

```

/* OUTPUT---

檔案寫入完成!!

-----*/





Reading Data from Binary Files (Buffered)

```
01  /* 讀取二進位檔案的內容 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      double a,b;
07      int i,arr[3];
08      FILE *fptr;
09      fptr=fopen("c:\\prog\\number.bin","rb"); /* 開啟檔案 */
10      fread(&a,sizeof(double),1,fptr); /* 把讀取的資料設定給 a 存放 */
11      fread(&b,sizeof(double),1,fptr); /* 把讀取的資料設定給 b 存放 */
12      fread(arr,sizeof(int),3,fptr); /* 把讀取的資料設定給陣列 arr 存放 */
13
14      printf("a=%4.2f\n",a);
15      printf("b=%4.2f\n",b);
16      for(i=0;i<3;i++)
17          printf("arr[%d]=%d\n",i,arr[i]);
18      fclose(fptr); /* 關閉檔案 */
19      system("pause");
20      return 0;
21 }
```

/* OUTPUT---

```
a=3.14
b=6.28
arr[0]=12
arr[1]=43
arr[2]=64
```

***/**



Writing Data to Binary Files (Non-buffered) - write()

```
01 /* 輸入資料到二進位檔案 */
```

```
02 #include <stdio.h>
```

```
03 #include <stdlib.h>
```

```
04 #include <fcntl.h>
```

```
05 #include <io.h>
```

```
06 #include <sys/stat.h>
```

```
07 int main(void)
```

```
08 {
```

```
09     int f1;
```

```
10     struct data          /* 定義結構 data */
```

```
11     {
```

```
12         char name[10];
```

```
13         int math;
```

```
14     }student={"Jenny",96};
```

```
15
```

```
/* OUTPUT--
```

```
資料已寫入檔案!!
```

```
*/
```

```
16     f1=open("c:\\prog\\score.bin",O_CREAT|O_WRONLY|O_BINARY,S_IREAD);
```

```
17     if((f1!=-1)          /* 檔案開啟成功 */)
```

```
18     {
```

```
19         write(f1,&student,sizeof(student));
```

```
20         close(f1);
```

```
21         printf("資料已寫入檔案!!\n");
```

```
22     }
```

```
23     else
```

```
24         printf("檔案開啟失敗!!\n");
```

```
25
```

```
26     system("pause");
```

```
27     return 0;
```

```
28 }
```



Reading Data to Binary Files (Non-buffered) - read()

```
01 /* 讀取二進位檔案的內容 */
02 #include <stdio.h>
03 #include <stdlib.h>
04 #include <fcntl.h>
05 #include <io.h>
06 #include <sys/stat.h>
07 int main(void)
08 {
09     int fl;
10     struct data
11     {
12         char name[10];
13         int math;
14     }student;          /* 宣告結構變數 student */
15     fl=open("c:\\prog\\score.bin",O_RDONLY | O_BINARY);
16
17     if((fl!=-1))      /* 檔案開啟成功 */
18     {
19         read(fl,&student,sizeof(student));/* 讀取資料並給 student 存放*/
20         printf("student.name=%s\n",student.name);
21         printf("student.math=%d\n",student.math);
22         close(fl);
23     }
24     else /* 檔案開啟失敗 */
25         printf("檔案開啟失敗!!\n");
26
27     system("pause");
28     return 0;
29 }
```

```
/* OUTPUT--
student.name=Jenny
student.math=96
*/
```



Lab 13

- Let the user input the student id and grade until the student id is **-1**
 - Write the Information to a file called “**grade.txt**”, formatted as

```
student_id-1    grade-1
student_id-2    grade-2
student_id-3    grade-3
                ...
student_id-k    grade-k
```

- Read the file generated by the previous problem (“**grade.txt**”).
 - Print the student id and the grade if he/she is failed (grade < 60)
 - Compute and print the average grade (float) on *stdout* after reading all data.