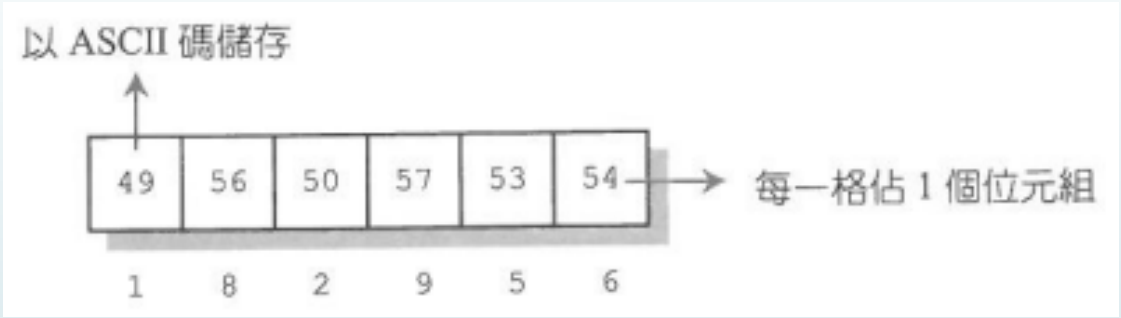
檔案處理

檔案

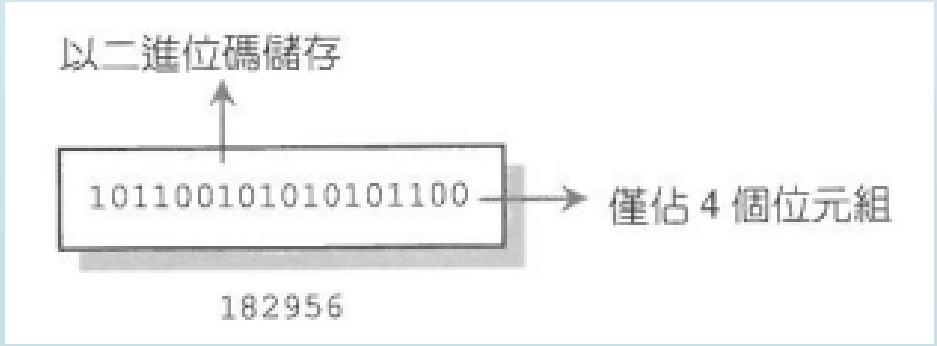
- 檔案依其目的分成三個檔案類型
  - 程式檔
    - 程式碼所存成的檔案
  - 執行檔
    - 編譯與連結過後的可執行檔案
  - 資料檔
    - 程式執行所產生或是程式執行時所需要的資料

文字檔與二進制檔案

- 文字檔
  - 以ASCII碼儲存，每個字元皆佔有一個位元組

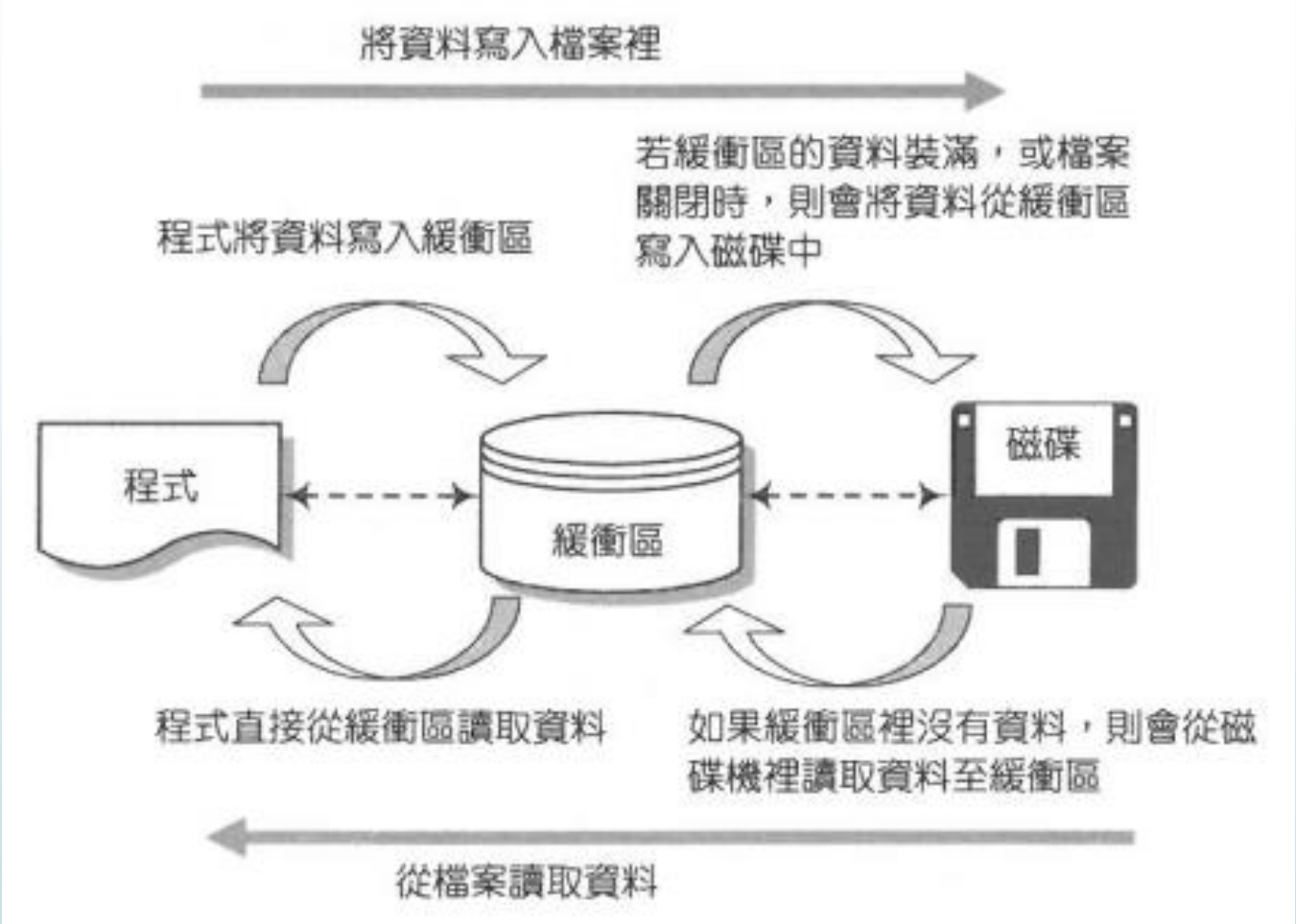


- 二進制檔案
  - 以二進位的格式儲存，如影像檔或執行檔
  - 以資料型態為長度為儲存單位
  - 在DEV C++中int佔有4個位元組
  - 相同資料下二進制檔會比文字檔小



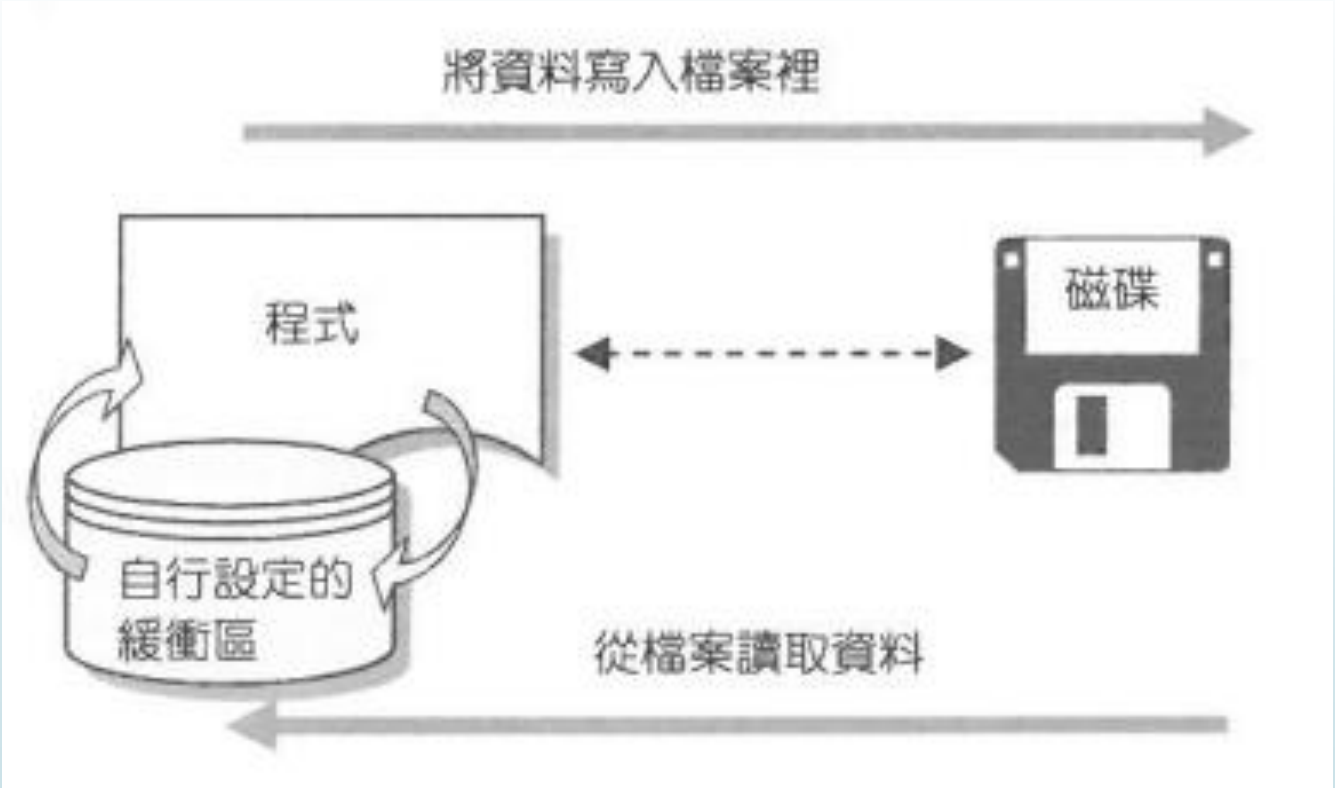
有緩衝區的檔案處理

- 有緩衝區(buffer)檔案處理
  - 讀取資料
    - 先到緩衝區讀取資料，如果緩衝區沒有資料，則會從資料檔讀取資料至緩衝區後，再由緩衝區把資料讀取至程式中
  - 寫入資料
    - 先寫入至緩衝區中，待緩衝區內資料裝滿或檔案關閉時，在寫入資料檔中



無緩衝區的檔案處理

- 無緩衝區的檔案處理



檔案處理步驟

- 無論使用者使用何種性質的檔案處理函數，其步驟如下所示
  - 1.開啟檔案
    - 將欲新增或修改的檔案開啟
  - 2.更新檔案內容
    - 將新資料寫入檔案中
  - 3.關閉檔案
    - 檔案使用完畢，要將檔案關閉才能確保資料全部寫入檔案

有緩衝區的檔案處理函數

- 有緩衝區的檔案處理
  - 好處在於不需要不斷的做磁碟的輸入與輸出，可增加程式執行效率
  - 缺點是必須佔用一塊記憶體空間，如果沒有關閉檔案或是系統當機，會因為留在緩衝區的資料尚未寫入到磁碟而造成資料的流失
  - 原型定義在stdio.h標頭檔中
  - 開啟檔案前必須先宣告一個指向檔案的指標，這個指標可以在開啟檔案後，記錄這個檔案所使用的緩衝區之起始位置
  - 宣告完成後，需要將指標變數指向某個檔案，待開啟檔案後，這個指標變數即代表某個被指向的檔案
  - 語法

```
FILE *指標變數;    /* 宣告指向檔案的指標 */
```

fopen()參數

- fopen
  - 語法

```
fopen("欲開啟檔案名稱","存取模式");
```

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 fopen() 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

fopen()

- 若開啟失敗會回傳指標NULL
  - NULL被定義在stdio.h裡的一個指標，指標被設為NULL表示它不指向任何變數
- 若開啟成功，則會回傳一個指向該檔案的指標

```
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 */
}
```

- 指定到資料夾時需加上「\」

```
fptr=fopen("c:\\prog\\abc.txt","r");   /* 開啟 c:\prog 資料夾下的檔案 */
```

檔案處裡函數的整理

函數功能	格式及說明
開啟檔案	`FILE *fopen(const char *filename, const char *mode);` 開啟指定的檔案，並指定存取模式。fopen() 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。fopen() 的傳回值為檔案指標，開檔失敗傳回 NULL
關閉檔案	`int fclose(FILE *fptr);` 關閉由 fptr 所指向的檔案，關檔成功傳回 0
讀取字元	`int getc(FILE *fptr);` 由 fptr 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	`int putc(int ch,FILE *fptr);` 將字元 ch 寫入由 fptr 所指向的檔案
讀取字串	`char *fgets(char *str,int maxchar,FILE *fptr);` 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
寫入字串	`int fputs(const char *str,FILE *fptr);` 將字串 str 寫入 fptr 所指向的檔案
檢查檔案是否結束	`int feof(FILE *fptr);` 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

函數功能	格式及說明
區塊輸入	`size_t fread(void *p,size_t s,size_t cnt,FILE *fptr);` 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
區塊輸出	`size_t fwrite(const void *p,size_t s,size_t cnt,FILE *fptr);` 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

處理檔案注意事項

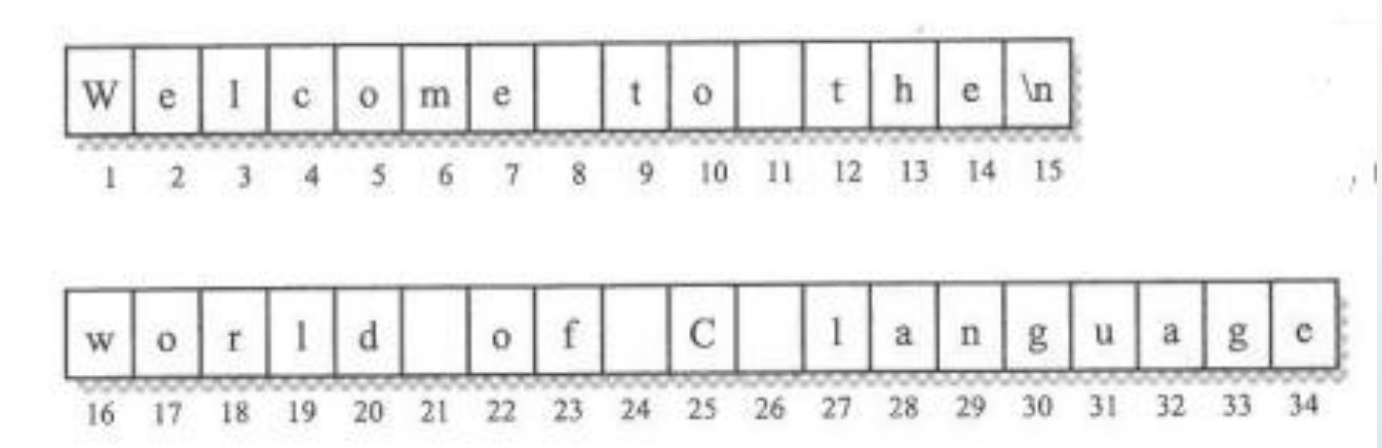
- 當檔案處裡完成時，切記一定要裡用fclose()將檔案關閉
  - 緩衝區的資料，才不會因程式結束而沒有寫入檔案
  - 釋放這個檔案所佔用的記憶體空間，以供其他檔案使用，檔案所占用的記憶體區域包含緩衝區及檔案的結構

範例-讀取檔案

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
 {
   FILE *fptr;       /* 宣告指向檔案的指標fptr */
   char ch;          /* 宣告字元變數ch，用來接收讀取的字元 */
   int count=0;      /* 宣告整數count，用來計算檔案的字元數*/

   fptr=fopen("welcome.txt","r");        /* 開啟檔案 */
   if(fptr!=NULL) /* 如果fopen()的傳回值不為NULL，代表檔案開啟成功 */
   {
      while((ch=getc(fptr))!=EOF)    /* 判斷是否到達檔尾 */
      {
         printf("%c",ch);            /* 一次印出一個字元 */
         count++;
      }
      fclose(fptr);                 /* 關閉所開啟的檔案 */
      printf("\n總共有%d個字元\n",count);
   }
   else          /* 檔案開啟失敗 */
      printf("檔案開啟失敗!!\n");

   system("pause");
   return 0;
}
```

範例詳解

- 空白字元及換行字元都會被計算在內



- **EOF**
  - **是C語言的關鍵字，定義在stdio.h中一個整數值(其值為 -1)**
  - **代表檔案結尾(end of file , EOF)**

範例-複製檔案內容至其他檔案

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   FILE *fptr1,*fptr2;      /* 宣告指向檔案的指標fpt1與fpt2 */
   char ch;

   fptr1=fopen("13_welcome.txt","r");    /* 開啟可讀取的檔案 */
   fptr2=fopen("13_output.txt","w");     /* 開啟可寫入的檔案 */

   if((fptr1!=NULL) && (fptr2!=NULL))        /* 如果開檔成功 */
   {
      while((ch=getc(fptr1))!=EOF)           /* 判斷是否到達檔尾 */
         putc(ch,fptr2);          /* 將字元ch寫到fptr2所指向的檔案 */
      fclose(fptr1);              /* 關閉fptr1所指向的檔案 */
      fclose(fptr2);              /* 關閉fptr2所指向的檔案 */
      printf("檔案拷貝完成!!\n");
   }
   else
      printf("檔案開啟失敗!!\n");

   system("pause");
   return 0;
}
```

範例-鍵盤輸入字串附加在檔案中

```c
#include <stdio.h>
#include <conio.h>       /* 函數getche()的原型定義在這兒 */
#include <stdlib.h>
#define ENTER 13          /* Enter鍵的ASCII碼為13 */
#define MAX 80
int main(void)
{
   FILE *fptr;
   char str[MAX],ch;     /* 宣告字元陣列str，用來儲存由鍵盤輸入的字串 */
   int i=0;
   fptr=fopen("14_output.txt","a");

   printf("請輸入字串，按ENTER鍵結束輸入:\n");
   while((ch=getche())!=ENTER && i<MAX)  /* 按下的鍵不是ENTER且i<MAX */
      str[i++]=ch;       /* 一次增加一個字元到字元陣列str中 */

   putc('\n',fptr);      /* 寫入換行字元 */
   fwrite(str,sizeof(char),i,fptr);   /* 寫入字元陣列str */
   fclose(fptr);         /* 關閉檔案 */
   printf("\n檔案附加完成!!\n");

   system("pause");
   return 0;
}
```

範例-使用fread讀取檔案

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 80
int main(void)
{
   FILE *fptr;
   char str[MAX];
   int bytes;                  /* 存放fread()成功讀取的字元數 */
   fptr=fopen("15_output.txt","r");

   while(!feof(fptr))          /* 如果還沒讀到檔尾  */
   {
      bytes=fread(str,sizeof(char),MAX,fptr);
      if(bytes<MAX)
         str[bytes]='\0';
      printf("%s\n",str);     /* 印出檔案內容 */
   }
   fclose(fptr);     /* 關閉檔案 */

   system("pause");
   return 0;
}
```

無緩衝區的檔案處理函數

- 無緩衝區的檔案處理
  - 資料存取直接透過磁碟，而不透過緩衝區
  - 不用佔用一大塊記憶體空間當緩衝區
  - 資料寫入檔案動作時，馬上寫入到磁碟中，如果系統忽然當機，所受到的損失較小
  - 缺點是由於磁碟運轉的速度較慢，在讀取或寫入資料時容易拖累程式執行的速度，也應此程式設計師通常都會自行設置一塊記憶體(如陣列)當成緩衝區
  - 標頭檔
    - fcntl.h (file control)及io.h (input/ouput)
    - 設定檔案屬性的常數定義是放在sys/stat.h

```
#include <fcntl.h>
#include <io.h>
#include <sys/stat.h>
```

open()

- open()
  - 語法

```
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

開啟模式

- 開啟模式
  - 基本模式 (只能選一個)
  - 修飾模式 (可以選多個，也可以不使用)
    - 使用位元運算子OR「|」將所有的模式串起

```
O_WRONLY                  /* 開啟舊檔，此檔只供寫入，不能讀取 */
O_WRONLY|O_APPEND         /* 開啟舊檔，此檔可以附加資料，但不能讀取 */
O_WRONLY|O_CREAT|O_APPEND /* 開啟舊檔，如不存在，則建立新檔，並可附加資料 */
O_RDONLY|O_TEXT           /* 開啟已存在的文字檔，且只供讀取 */
```

存取屬性

- 存取屬性
  - 檔開始檔案模式為 O_CREATE 時，必須寫出該檔案的存取屬性
  - 選用參數

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

判斷開啟檔案

- 必須宣告一個整數變數來接收open()的回傳值
  - 開啟檔案失敗回傳 -1
  - 開啟檔案成功回傳一個整數值，此數值稱為檔案代號「handle」
  - 範例

```
int f1;
f1=open("c:\\prog\\abc.txt",O_WRNDLY|O_CREAT|O_TEXT,S_IREAD);
```

無緩衝區的檔案處理函數1

函數功能	格式
開啟檔案	`int open(const char *filename,int oflag[,int pmode]);` 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。oflag 代表開檔模式，oflag 之後的中括號所包圍的引數 pmode（代表存取屬性）為可有可無，視檔案的需要而取捨
關閉檔案	`int close(int handle);` 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	`int creat(const char *filename,int pmode);` 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

無緩衝區的檔案處理函數2

函數功能	格式
讀取資料	`int read(int handle,char *buffer,unsigned count);` 讀取檔案中的資料，最多可一次讀取 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	`int write(int handle,char *buffer,unsigned count);` 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

範例-複製檔案內容

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <io.h>
#include <sys/stat.h>
#define SIZE 512        /* 設定read()一次可讀取的最大位元組為512 */
int main(void)
{
   char buffer[SIZE];
   int f1,f2,bytes;

   f1=open("23_welcome.txt",O_RDONLY|O_TEXT);
   f2=creat("23_output.txt",S_IWRITE);

   if((f1!=-1)&&(f2!=-1))        /* 測試檔案是否開啟成功 */
   {
      while(!eof(f1))            /*  如果還沒有讀到檔案末端 */
      {
         bytes=read(f1,buffer,SIZE);     /* 從f1讀取資料 */
         write(f2,buffer,bytes);         /* 將資料寫入檔案f1中 */
      }
      close(f1);
      close(f2);
      printf("檔案拷貝完成!!\n");
   }
   else
      printf("檔案開啟失敗!!\n");

   system("pause");
   return 0;
}
```

有緩衝區函數處理二進制檔案

- fopen()需指明檔案為二進制

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

- 範例

```
FILE *fptr;                          /* 宣告 fptr 為一指向檔案的指標變數 */
fptr=fopen("test.bin","ab");  /* 開啟可供附加資料的二進位檔案 test.bin */
```

範例-輸入資料到二進制檔案

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    double a=3.14,b=6.28;
    int arr[]={12,43,64};
    FILE *fptr;

    fptr=fopen("number.bin","wb");    /* 開啟檔案 */
    fwrite(&a,sizeof(double),1,fptr);     /* 寫入變數a的值 */
    fwrite(&b,sizeof(double),1,fptr);     /* 寫入變數b的值 */
    fwrite(arr,sizeof(int),3,fptr);  /* 寫入陣列arr的所有元素 */

    fclose(fptr);          /* 關閉檔案 */
    printf("檔案寫入完成!!\n");

    system("pause");
    return 0;
}
```

範例-讀取二進制檔案內容

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   double a,b;
   int i,arr[3];
   FILE *fptr;

   fptr=fopen("26_number.bin","rb");       /* 開啟檔案 */
   fread(&a,sizeof(double),1,fptr);  /* 把讀取的資料設定給a存放 */
   fread(&b,sizeof(double),1,fptr);  /* 把讀取的資料設定給b存放 */
   fread(arr,sizeof(int),3,fptr); /* 把讀取的資料設定給陣列arr存放 */

   printf("a=%4.2f\n",a);
   printf("b=%4.2f\n",b);
   for(i=0;i<3;i++)
      printf("arr[%d]=%d\n",i,arr[i]);

   fclose(fptr);          /* 關閉檔案 */

   system("pause");
   return 0;
}
```

無緩衝區函數處理二進制檔案

- 範例

```
int f1;             /* 宣告整數變數 f1 來接收檔案代號 */
f1=open("test.bin", O_WRONLY|O_CREAT|O_BINARY, S_IREAD);
```

- 開啟的檔案只供寫入
- 如果檔案不存在，則會開啟一個新檔
- 寫入的格式為二進制
- 設定檔案屬性為唯讀，下次開啟檔案時，僅供讀取，不能寫入資料

範例-無緩衝區函數寫入二進制檔案

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <io.h>
#include <sys/stat.h>
int main(void)
{
    int f1;
    struct data                 /* 定義結構data */
    {
        char name[10];
        int math;
    }student={"Jenny",96};          /* 宣告結構變數data，並設定初值 */

    f1=open("28_score.bin",O_CREAT|O_WRONLY|O_BINARY,S_IREAD);
    if((f1!=-1))        /* 檔案開啟成功 */
    {
        write(f1,&student,sizeof(student));
        close(f1);
        printf("資料已寫入檔案!!\n");
    }
    else
        printf("檔案開啟失敗!!\n");

    system("pause");
    return 0;
}
```

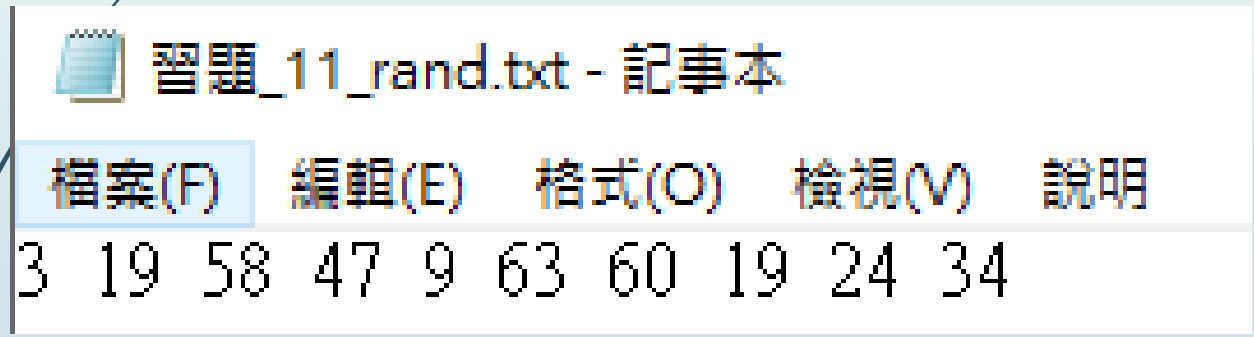
範例-無緩衝區函數讀取二進制檔案

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <io.h>
#include <sys/stat.h>
int main(void)
{
   int f1;
   struct data
   {
      char name[10];
      int math;
   }student;         /* 宣告結構變數student */
   f1=open("29_score.bin",O_RDONLY | O_BINARY);

   if((f1!=-1))      /* 檔案開啟成功 */
   {
      read(f1,&student,sizeof(student)); /* 讀取資料並給student存放 */
      printf("student.name=%s\n",student.name);
      printf("student.math=%d\n",student.math);
      close(f1);
   }
   else /* 檔案開啟失敗 */
      printf("檔案開啟失敗!!\n");

   system("pause");
   return 0;
}
```

習題

- 產生10個 1~64之間的整數亂數，並將其寫入到「rand.txt」檔案中

```
亂數為:  3  19  58  47   9  63  60  19  24  34
檔案寫入完成!!
請按任意鍵繼續 . . .
```



- 整數寫入到檔案方法
  - fprint(檔案變數 ,資料格式 , …);

隨機函數

- rand()
  - 隨機產生 0 ~ RAND_MAX 之間的值
  - 可使用printf(”%d”,RAND_MAX)觀察
  - 由於電腦實際上並沒有辦法自己產生「真正的亂數」，只能透過複雜的數學演算法模擬出類似亂數的數值資料，而在模擬亂數時，需要設定一個亂數種子，電腦會根據這個亂數種子來計算出一連串的亂數，相同的亂數種子就會產生相同的亂數序列，所以如果要讓產生的亂數每次都不同，就要設定不同的亂數種子。
- srand()
  - 設定亂數種子
  - 範例

```
srand( time(NULL) );
```

隨機函數應用

- 隨機產生固定範圍的整數

```
int min = 4;
int max = 10;

/* 產生 [min , max] 的整數亂數 */
int x = rand() % (max - min + 1) + min;
```

- 隨機產生0~1之間的浮點數

```
double x = (double) rand() / (RAND_MAX + 1.0);
```

- 隨機產生特定範圍的浮點數

```
double min = 3.6;
double max = 7.8;

/* 產生 [min , max) 的浮點數亂數 */
double x = (max - min) * rand() / (RAND_MAX + 1.0) + min;
```