ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Computer Programming II
การเขียนโปรแกรมคอมพิวเตอร์2
LECTURE#10 แฟ้มข้อมูล(File)

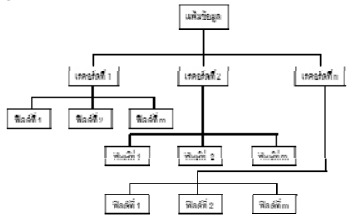
อ.สถิตย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

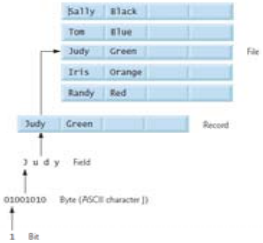
KMUTNB

ความหมายของแฟ้มข้อมูลในภาษา C

- แฟ้มข้อมูล (data file) คือ แฟ้มที่มีการเก็บข้อมูลที่มีความสัมพันธ์กันมาไว้ด้วยกัน โดยมีการเก็บข้อมูลอย่างต่อเนื่องกันไป ตั้งแต่ต้นแฟ้มข้อมูลไปจนกระทั่งจบแฟ้มข้อมูล
- ผู้เขียนข้อมูลสามารถแบ่งข้อมูลที่ต้องการจัดเก็บลงในแฟ้มเป็น field หรือ record ก็ได้ หรืออาจจัดเก็บข้อมูลตามแนวขนาดเนื้อที่โดยไม่จำเป็นต้องแบ่งข้อมูลในแฟ้มเป็น field หรือ record ก็ได้
- โดยปกติผู้เขียนโปรแกรมภาษา C นิยมแบ่งข้อมูลที่ต้องการลงในแฟ้มเป็น field หรือ record เพราะมีความสะดวกในการเรียกใช้ข้อมูลจากแฟ้มที่ต้องการ



ลักษณะโครงสร้างของแฟ้มข้อมูลทั่วไป



ประเภทของแฟ้มข้อมูล

- Text file เป็นไฟล์ที่เก็บข้อมูลในรูปแบบของตัวอักษรและทำการแยกแต่ละบรรทัดของ text file ออกจากกันด้วย
- Binary file เป็นไฟล์ที่เก็บข้อมูลในรูปแบบเฉพาะของคอมพิวเตอร์

การประมวลผลแฟ้มข้อมูลในภาษา C

- โดยปกติแล้วผู้เขียนโปรแกรมเกี่ยวกับแฟ้มข้อมูลในภาษา C จะมีความต้องประมวลผลแฟ้มข้อมูลอยู่ 3 แบบ คือ
  1) การบันทึกข้อมูลในแฟ้มข้อมูล (write data into file)
  2) การอ่านข้อมูลขึ้นจากแฟ้มข้อมูลขึ้นมาใช้งาน (read data from file)
  3) การเพิ่มข้อมูลลงไปในแฟ้มข้อมูล (append data into file)

ขั้นตอน	บันทึกข้อมูลเก็บไว้ในแฟ้ม	อ่านข้อมูลขึ้นจากแฟ้ม	เพิ่มข้อมูลในแฟ้ม
เปิดแฟ้มข้อมูลด้วยคำสั่ง fopen( ) ตั้งชื่อแฟ้มข้อมูล (file name) พร้อมกับระบุ mode	fopen(“ ”,“w”)	fopen(“ ”,“r”)	fopen(“ ”,“a”)
ข้อมูลที่ต้องการเป็นตัวอักขระตัวเดียว (single character)	putc( )	getc( )	fprintf( )
ข้อมูลที่ต้องการบันทึกเป็นตัวเลขจำนวนเต็ม (integer) หรือตัวเลขจำนวนทศนิยม (floationg point) หรือสตริง (strings)	fprintf( )	fscanf( )	fprintf( )
ข้อมูลที่ต้องการบันทึกเป็นข้อมูลแบบโครงสร้าง (structures) หรือตัวแปรชุด (arrays)	fwrite( )	fread( )	fwrite( )
ปิดแฟ้มข้อมูลทุกครั้งเพื่อป้องกันความเสียหายที่อาจเกิดขึ้นได้	fclose( )	fclose( )	fclose( )
ข้อควรระวัง	mode “w” เป็นการเปิดแฟ้มข้อมูลเพื่อบันทึกข้อมูลลงในแฟ้มเท่านั้น ถ้าเป็นแฟ้มข้อมูลเก่าจะมีผลทำให้ข้อมูลทั้งหมดในแฟ้มข้อมูลเก่า ถูกลบทิ้งไปโดยอัตโนมัติ	mode “r” เป็นการเปิดแฟ้มข้อมูล เพื่ออ่านข้อมูลขึ้นจากแฟ้มข้อมูลอย่างเดียว ไม่สามารถบันทึกข้อมูลเพิ่มเติม	mode “a” เป็นการเปิดแฟ้มข้อมูลสำหรับบันทึกข้อมูลเพิ่มเติมลงไปในแฟ้ม

File pointer

- File Pointer คือ pointer ที่ชี้ไปยัง data type FILE ซึ่งถูกกำหนดไว้ใน stdio.h โดยเป็น structure ที่เป็นที่เก็บรวบรวมข้อมูลที่เกี่ยวข้องกับ file นั้น
  - File descriptor
  - แหน่งปัจจุบันใน buffer
  - Pointers ไปยัง buffers
  - ในขณะนั้น file กำลังถูก read หรือ write อยู่หรือไม่
- ก่อนที่ file จะสามารถถูก read หรือ write จะต้องใช้ฟังก์ชันใน C library ชื่อ fopen() เปิด file ไว้ก่อน ซึ่งมี limit ว่าจะสามารถเปิดได้กี่ files พร้อมๆ กัน
- หลังจากที่ files ถูกใช้เสร็จแล้ว files จะต้องถูกปิดโดยใช้ฟังก์ชัน fclose() ซึ่งเป็นการเคลียร์ file buffer และตัด connection ไปยัง file นั้นออก

การเปิดไฟล์ (Opening Files)

- ใช้ฟังก์ชัน fopen() ซึ่งมีรูปแบบดังนี้

```
FILE *fopen( char *path, char *mode);
```

- ฟังก์ชันนี้มี 2 arguments ทั้ง 2 arguments เป็น pointer ไปยัง character strings โดย pointer แรกจะเก็บค่า address ของชื่อ file ที่จะถูกเปิด และ pointer ที่ 2 เป็น address ของ access mode
- ตัวอย่าง

```
FILE *fptr;
fptr=fopen(“song.txt”, “w”);
```

ความหมายของ access mode

Content ของaccess mode string	ความหมาย
"w"	เปิด file เพื่อ write ซึ่งถ้ามี file นั้นอยู่แล้วการเลือกใช้ mode นี้จะทำการลบ content ของเดิมออก
"r"	เปิด file เพื่อ read
"a"	เปิด file เพื่อ write ถ้ามี file เดิมอยู่ จะ write ข้อมูลต่อท้าย file เดิม
“r+"	เปิด file เพื่ออ่านและเขียน
“w+"	เปิด file ใหม่เพื่ออ่านและเขียน
"a+"	เปิด file สำหรับอ่านและเขียนต่อท้าย

ถ้าฟังก์ชัน fopen() ทำงานสำเร็จจะ return ค่าของ file pointer แต่หากทำงานไม่สำเร็จ (ไม่สามารถเปิด file ได้) ฟังก์ชันจะ return NULL

การปิดแฟ้ม (Closing files)

- เมื่อใช้ files แล้วจะต้องปิดด้วยฟังก์ชัน `flose()` ซึ่งมีรูปแบบดังนี้

```
int fclose( FILE *stream);
```

- ฟังก์ชัน `flose()` จะ return ค่า 0 หากทำงานสำเร็จ หากทำไม่สำเร็จจะ return ค่าเป็น EOF
- ตัวอย่าง

```
FILE *fptr;
fptr=fopen(“song.txt”,“r”);
…
fclose(fptr);
```

การใช้ fopen() และ fclose()

```
#include <stdio.h>
#include <conio.h>
int main(void) {
   FILE *fptr;
   fptr = fopen("D:\\ascii.txt","r");
   if(fptr != NULL){
     printf("I can open file\n");
     fclose(fptr);
   } else
     printf("Error! File can’t be opened\n");
   getch();
   return (0);
 }
```

การอ่านและเขียนตัวอักษรจากแฟ้ม

- ในการอ่าน character จาก input stream ที่ถูกชี้โดย file pointer สามารถใช้ฟังก์ชัน getc() และ fgetc() ซึ่งมีรูปแบบของฟังก์ชัน ดังนี้

```
char fgetc(FILE *stream);
char getc(FILE *stream);
```

- โดยหากฟังก์ชันทำงานสำเร็จ ฟังก์ชันจะคืนค่าเป็น character ในรูปแบบของ char และหากทำงานไม่สำเร็จจะคืนค่า EOF กลับมา ขอให้ดูวิธีการใช้ฟังก์ชันจากโปรแกรมต่อไปนี้

การอ่านและเขียนตัวอักษรจากแฟ้ม (2)

```
#include <stdio.h>
#include <conio.h>
int main() {
   char ch;
   FILE *fptr;
   fptr = fopen("D:\\ascii.txt","r");
   if(fptr != NULL){
      printf("I can successfully open my file\n");
      while ((ch = getc(fptr))!=EOF){
      printf("%c",ch);  }
      fclose(fptr);
   } else{
      printf("Error! I can't open my file\n");
   }
  getch();
  return 0;
}
```

การอ่านและเขียนตัวอักษรจากแฟ้ม (3)

- ฟังก์ชัน **getc( )**
  เป็นฟังก์ชันที่ใช้อ่านข้อมูลตัวอักขระตัวเดียวขึ้นจากแฟ้มข้อมูลที่ต้องการ
- **รูปแบบการใช้ฟังก์ชัน**
- `getc(fp);`
  หรือ
  `single_char = getc(fp);`
- โดยที่
  `single_char` คือ ตัวแปรชนิด `single character` ที่ใช้เก็บข้อมูลตัวอักขระ ตัวเดียว
  `fp` คือ `file pointer` ของแฟ้มข้อมูลที่ต้องการบันทึกข้อมูลลงไป

การอ่านและเขียนตัวอักษรจากแฟ้ม (4)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main(void)
{
     FILE *fp;
     char ch;
     if(( fp=fopen("D:\\file1.txt","r"))==NULL)
     {
          printf("Error in open file");
          printf("\007");
          exit(1);
     }
     do
     {
          ch=getc(fp);
          putchar(ch);
     } while(ch !=EOF);
     fclose(fp);
     getch();
     return 0;
}
```

ฟังก์ชันสำหรับการเขียนตัวอักษรไปสู่แฟ้ม

- ฟังก์ชันสำหรับการ write character ไปสู่ file 2 ฟังก์ชันดังนี้

```
int fputc(int c, FILE *stream);
int putc(int c, FILE *stream);
```

- ฟังก์ชันทั้ง 2 มี 2 arguments โดย argument แรกเป็น character ในรูปแบบของ integer ส่วน argument ที่ 2 เป็น file pointer
- ถ้าฟังก์ชันทำงานสำเร็จจะ return character ที่ output ไปในรูปแบบของ integer หากทำงานไม่สำเร็จจะคืนค่า EOF

การ copy file

```
#include <stdio.h>
#include <conio.h>
int  main() {
   char ch;
   FILE *infile, *outfile;
   infile = fopen("D:\\file1.txt","r");
   outfile = fopen("D:\\targetFile.txt","w");
   if((infile == NULL)||(outfile == NULL))
   {
      printf("File open error\n");
   }
   else {
      ch = fgetc(infile);
      while (ch != EOF) {
          fputc((char)ch, outfile);
          ch = fgetc(infile);
      }
   }
   if (infile != NULL) fclose(infile);
   if (outfile != NULL) fclose(outfile);
   getch();
   return 0;
}
```

ฟังก์ชันสำหรับการเขียนตัวอักษรไปสู่แฟ้ม (2)

- **ฟังก์ชัน `putc( )`** เป็นฟังชันที่ใช้บันทึกข้อมูลตัวอักขระตัวเดียวลงไปในแฟ้มข้อมูลที่ต้องการ
- **รูปแบบการใช้ฟังก์ชัน**
  - `putc(single_char,fp);`
- โดยที่
  - **`single_char`** คือ ค่าคงที่ชนิดตัวอักขระตัวเดียว หรือตัวแปรชนิด `single_character`
  - **`fp`** คือ `file pointer` ของแฟ้มข้อมูลที่ต้องการบันทึกข้อมูลลงไป

ฟังก์ชันสำหรับการเขียนตัวอักษรไปสู่แฟ้ม (3)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main(void)
{
     FILE *fp;
     char ch;
     if(( fp=fopen("D:\\file1.txt","w"))==NULL)
     {
          printf("Error in open file");
          printf("\007");
          exit(1);
     }
     printf("Please press <Enter> to quit program.\n");
     printf("\nEnter your sentence : ");
     do {
         ch=getche( );
         putc(ch, fp);
         } while(ch !='\r');
     fclose(fp);
     getch();
     return 0;
}
```

การอ่านและเขียน String ลงแฟ้ม

มีฟังก์ชัน ที่เกี่ยวข้องดังต่อไปนี้

```
int fputs(const char *s, FILE *stream);

char *fgets(char *s, int size, FILE *stream);
```

การอ่านและเขียน String ลงแฟ้ม (2)

ใช้ฟังก์ชัน fprintf() และ fscanf() ซึ่งทำงานเหมือนกับ printf() และ scanf() ยกเว้นมี arguments ตัวแรกเป็น file pointer รูปแบบของฟังก์ชันมีดังนี้

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

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

การอ่านและเขียน String ลงแฟ้ม (3)

```
#include <stdio.h>
#include <conio.h>
#define STRSIZE 1000
#define READ "r"
int main() {
  FILE *fp;
  int wc = 0;
  int checkread;
  char word[STRSIZE];
  fp = fopen("D:\\file1.txt",READ);
  if (fp == NULL)
    fprintf(stderr, "Can't open file");
  else {
    checkread = fscanf(fp,"%s",word);
    while (checkread != EOF) {
       wc++;
       checkread = fscanf(fp,"%s",word);
    }
     fprintf(stdout,"Number of words %d\n",wc);
     fclose(fp);
   }
   getch();
  return 0;
}
```

การอ่านและเขียน String ลงแฟ้ม (4)

- ฟังก์ชัน **fprintf( )**
  เป็นฟังก์ชันที่ใช้บันทึกข้อมูล (write) ลงแฟ้มโดยสามารถจัดรูปแบบข้อมูลที่ต้องการบันทึกได้
  - คล้ายกับฟังก์ชัน printf( )
  - แตกต่างกันตรงที่ printf( )เป็นฟังก์ชันที่ใช้พิมพ์ผลลัพธ์ออกทางจอภาพแต่ฟังก์ชัน fprintf( ) ใช้บันทึกข้อมูลลงแฟ้ม
- รูปแบบการใช้ fprintf(fp,control string,variable list);
- โดยที่
  - fp คือ ตัวชี้ตำแหน่งในแฟ้มข้อมูล
  - control string คือ รหัสรูปแบบข้อมูลและรหัสควบคุมใช้เหมือนฟังก์ชัน printf( ) เช่น สามารถระบุชนิดของข้อมูลที่ต้องการบันทึกลงแฟ้มเป็น %d, %c, %f, %s หรือใช้รหัสควบคุม \n หรือ \t ก็ได้
  - variable list คือ ค่าคงที่ ตัวแปร หรือนิพจน์ที่จะเขียนลงแฟ้มข้อมูล ถ้าเป็นค่าคงที่สตริงต้องเขียนอยู่ภายในเครื่องหมาย “…”

การอ่านและเขียน String ลงแฟ้ม (5)

- **ตัวอย่าง** แสดงการใช้ฟังก์ชัน fprintf( )

```
int x=15; float f=4.562; char str[20]= “Computer”;
FILE *fptr;
fprintf(fptr, “%d \t %f \t %s \n”,x,f,str);
```

- โดยที่
  fptr คือ ตัวชี้ตำแหน่งในแฟ้มข้อมูล (file pointer)
  x คือ ตัวแปร int เก็บข้อมูลตัวเลข 15 ซึ่งจะถูกบันทึกลงแฟ้มเป็น field ที่ 1
  f คือ ตัวแปร float เก็บข้อมูลตัวเลข 4.562 ซึ่งจะถูกบันทึกลงแฟ้มเป็น field ที่ 2
  str คือ ตัวแปร string เก็บข้อความว่า “Computer” ซึ่งจะถูกบันทึกลงแฟ้มเป็น field ที่ 3
  \t คือ รหัสควบคุมที่สั่งให้ tab ไป 1 ครั้ง ก่อนที่จะบันทึกข้อมูล field ต่อไป
  \n คือ รหัส new line ใช้สั่งให้ขึ้นบรรทัดใหม่ในแฟ้มข้อมูล

การอ่านและเขียน String ลงแฟ้ม (6)

- ฟังก์ชัน fscanf( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูล (read) ขึ้นจากแฟ้มข้อมูลแล้วนำมาเก็บไว้ในตัวแปรที่ต้องการได้โดยมีการทำงานคล้ายกับฟังก์ชัน scanf( )
- รูปแบบการใช้

  fscanf(fp, control string, variable list);
- โดยที่
- fp คือ ตัวชี้ตำแหน่งในแฟ้มข้อมูล
  control string คือ รหัสรูปแบบข้อมูลเช่น ระบุชนิดของข้อมูลที่ต้องการอ่านข้อมูลจากแฟ้มเป็น %d,%c,%f,%s และสามารถใช้รหัส new line หรือ \n ได้
  variable list คือ ชื่อตัวแปรที่ใช้เก็บข้อมูลที่อ่านมาจากแฟ้มข้อมูล โดยจะต้องระบุเครื่องหมาย & (ampersand) นำหน้าชื่อตัวแปรด้วย ยกเว้นตัวแปรสตริงเท่านั้นที่ไม่ต้องมีเครื่องหมาย &

การอ่านและเขียน String ลงแฟ้ม (7)

- **ตัวอย่าง** แสดงการใช้ฟังก์ชัน fscanf( )

```
int i; float f; char str[80];
FILE *fptr;
fscanf( fptr,"%d %f %s \n",&i,&f,str);
```

- โดยที่

  i คือ ตัวแปรชนิด `int` ใช้เก็บข้อมูลที่อ่านจากแฟ้ม `field` ที่ 1

  f คือ ตัวแปรชนิด `float` ใช้เก็บข้อมูลที่อ่านจากแฟ้ม `field` ที่ 2

  str คือ ตัวแปรสตริง ใช้เก็บข้อมูลที่อ่านจากแฟ้ม `field` ที่ 3

  fptr คือ `file pointer` ใช้ชี้ตำแหน่งข้อมูลในแฟ้ม

การอ่านและเขียน String ลงแฟ้ม (8)

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int main(void)
{
     FILE *fptr;
     int x=5;
     float f=4.5;
     char str[80]="C and C++ Programming";
     char fname[80];
     printf("Enter your file name ( file format is *.txt ) :");
     gets(fname);
     if( (fptr=fopen(fname, "w")) == NULL )
     {
         printf("Error in open file ");
         printf("\007");
         exit(1);
      }
         fprintf(fptr, "%d \t %.2f \t %s\n", x, f, str);
         printf("Successful to write data to file");
         fclose(fptr);
         getch();
         return 0;
}
```

การอ่านและเขียน String ลงแฟ้ม (9)

- การทำงานของโปรแกรมตัวอย่างจะให้เปิดแฟ้มข้อมูลขึ้นมาเพื่อเขียน
- โดยให้ผู้ใช้งานตั้งชื่อแฟ้มเอง แต่ให้มีนามสกุล `.txt` เช่น `D:\\test.txt` คือ ตั้งชื่อแฟ้ม `test.txt` เก็บไว้ที่ไดร์ฟ `D`
- ถ้าเปิดไม่ได้ก็จะพิมพ์ข้อผิดพลาด
- ถ้าเปิดได้แล้วจะนำค่าของตัวแปร `x, f` และ ตัวแปร `str` เขียนลงในแฟ้มข้อมูลดังกล่าว `1 record` ตามคำสั่ง
  - ซึ่งการเขียนข้อมูลจะมีการเว้นช่องว่างในแต่ละฟิลด์ เนื่องจากมีการใช้รหัส `\t` และ `\n` ทำให้เกิดความสะดวกในการอ่านข้อมูลจากแฟ้มขึ้นมาใช้งานต่อไป เขียนเสร็จแล้วก็จะใช้ฟังก์ชัน `fclose( )` เพื่อปิดแฟ้มข้อมูล

การอ่านและเขียน String ลงแฟ้ม (10)

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int main(void)
{
     FILE *fptr;
     int x=1;
     float f=8.5;
     char str[80]="C and C++
  Programming";
     int j;
     char fname[80];

     printf("Enter your file name (
  file format is *.txt ) :");
  gets(fname);
```

```
if( (fptr=fopen(fname, "w")) ==
   NULL )
  {
   printf("Error in open file ");
   printf("\007");
   exit(1);
  }
for(j=1; j<=5; j++) {
   fprintf(fptr, "%d \t %.2f \t
   %s\n ", x, f, str);
   x= x+1;
   f= f+1.5;
  }
fclose(fptr);
getch();
return 0;
}
```

การอ่านและเขียน String ลงแฟ้ม (11)

- ฟังก์ชัน **fwrite( )**

  เป็นฟังก์ชันที่ใช้เก็บข้อมูล ลงแฟ้มโดยที่การเก็บข้อมูลแต่ละครั้ง สามารถกำหนดขนาดของข้อมูลที่ต้องการบันทึกได้

  รูปแบบการใช้ `fwrite(&var, size, n, fp);`

- โดยที่

  &var คือ ตำแหน่งของตัวแปรที่เก็บข้อมูลที่ต้องการนำไปเก็บไว้ในแฟ้ม

  size คือ ขนาดของข้อมูลที่ต้องการ `write` ลงแฟ้มในแต่ละครั้ง ซึ่งสามารถหาได้จากฟังก์ชัน `sizeof(data type);` หรือ `sizeof(variable name);` เช่น `sizeof(int);` หรือ `sizeof(j);`

  n คือ จำนวนครั้งที่ต้องการ `write` ข้อมูลลงแฟ้ม

  fp คือ ตัวชี้ตำแหน่งข้อมูลในแฟ้ม (`file pointer`)

- **ข้อควรจำ** ฟังก์ชัน `fwrite( )` จะให้ค่าเลขจำนวนเต็มเท่ากับ `n` (จำนวนครั้งที่ `write` ข้อมูลลงแฟ้ม) เมื่อไม่เกิดข้อผิดพลาดในการเขียนข้อมูล และจะให้ค่าน้อยกว่า `n` เมื่อมีข้อผิดพลาดเกิดขึ้น

การอ่านและเขียน Struct ลงแฟ้ม

- แสดงการใช้ฟังก์ชัน `fwrite( )`

1)

- ```
  FILE *fptr;
  double x=1.2345678912345;
  fwrite(&x,sizeof(double x),1,fptr);
  ```

  เป็นการบันทึกข้อมูลที่เก็บไว้ในตัวแปร `x` ลงแฟ้มข้อมูล โดยทำการบันทึก 1 ครั้ง (`n=1`) ขนาดของข้อมูลที่บันทึกลงแฟ้มในแต่ละครั้งมีขนาด 8 `bytes` ตามชนิดตัวแปร `x`

2)

- ```
  FILE *fptr; int j;
  int a[10]={10,20,30,40,50,60,70,80,90,100};
  for (j=0;j<10;j++)
       fwrite(&a[j],sizeof(int a[j]),1,fptr);
  ```

การอ่านและเขียน Struct ลงแฟ้ม (2)

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<ctype.h>
int main(void)
{  struct
     {      char name[30];
            char id[20];
            float gpa;
     }student;
   char numstr[80], fname[80];
   FILE *fptr;
   printf("Enter your file name (file format is *.txt) :");
   gets(fname);
   if( (fptr=fopen(fname, "wb")) == NULL )
     {
         printf("Error in open file");
         exit(1);
     }
```

การอ่านและเขียน Struct ลงแฟ้ม (3)

```
 do {
        printf("\nEnter student name :");
        gets( student.name );
        printf("Enter student id :");
        gets( student.id );
        printf("Enter student gpa :");
        gets( numstr );
        student.gpa = atof(numstr);
        fwrite(&student, sizeof(student), 1, fptr);
        printf("Add another student(y/n)?:");
    } while( tolower(getche()) == 'y' );
fclose(fptr);
getch();
return 0;
}
```

การอ่านและเขียน Struct ลงแฟ้ม (4)

- ฟังก์ชัน **fread( )**
  ฟังก์ชัน fread( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูลจากแฟ้ม โดยที่แต่ละครั้งสามารถกำหนดขนาด (size) ของข้อมูลที่ต้องการอ่านได้
- รูปแบบการใช้ fread(&var,size,n,fp); โดยที่
  - **&var** คือ ตำแหน่งของตัวแปรที่เก็บข้อมูลที่ต้องการนำไปเก็บไว้ในแฟ้ม
  - **size** คือ ขนาดของข้อมูลที่ต้องการ read ขึ้นจากแฟ้มในแต่ละครั้ง ซึ่งสามารถหาได้จากฟังก์ชัน sizeof(data type); หรือ sizeof(variable name); เช่น sizeof(int); หรือ sizeof(j);
  - **n** คือ จำนวนครั้งที่ต้องการ read ข้อจากแฟ้ม
  - **fp** คือ ตัวชี้ตำแหน่งข้อมูลในแฟ้ม (file pointer)
- ข้อควรจำ
  - ฟังก์ชัน fread( ) จะให้ค่าตัวเลขจำนวนเต็ม = n เมื่อไม่เกิดข้อผิดพลาดในการอ่านข้อมูล และจะให้ค่าเป็นศูนย์ (0 หรือ NULL)
  - เมื่อมีข้อผิดพลาดในการอ่านข้อมูลจากแฟ้มหรือสิ้นสุดไฟล์ (EOF)

การอ่านและเขียน Struct ลงแฟ้ม (5)

- ตัวอย่าง แสดงการใช้ฟังก์ชัน fread( ) เช่น

```
FILE *fptr;
double x;
fread(&x,sizeof(double x),1,fptr);
```

- เป็นการอ่านข้อมูลในแฟ้มข้อมูลมาเก็บไว้ที่ตัวแปร x โดยทำการอ่านข้อมูล 1 ครั้ง (n=1) ขนาดของข้อมูลที่อ่านจากแฟ้มในแต่ละครั้งมีขนาด 8 bytes ตามชนิดตัวแปร x

```
FILE *fptr;
int a[10]; int j;
for (j=0;j<10;j++)
fread(&a[ j],sizeof(int a[ j]),1,fptr);
```

- เป็นการอ่านข้อมูลในแฟ้มมาเก็บไว้ในตัวแปรชุด a โดยทำการอ่านข้อมูล 10 ครั้ง โดยที่ขนาดของข้อมูลที่อ่านจากแฟ้มในแต่ละครั้งมีขนาด 2 bytes ตามชนิดตัวแปร a[j]

การอ่านและเขียน Struct ลงแฟ้ม (6)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main(void)
{   struct {
            char name[30];
            char id[20];
            float gpa;
    } student;
    char fname[80];
    FILE *fptr;
    printf("Enter your file name
  (file format is *.txt) :");
    gets(fname);
    if( (fptr=fopen(fname, "rb")) ==
  NULL )
    {
        printf("Error in open file");
        exit(1);
    }
```

```
while( fread(&student,
   sizeof(student),1, fptr) == 1
   )
  {
   printf("\nName = %s\n",
   student.name);
   printf("Id = %s\n",
   student.id);
   printf("GPA = %.2f\n",
   student.gpa);
  }
fclose(fptr);
getch();
return 0;
}
```

ฟังก์ชันที่ใช้ควบคุมตำแหน่งของ file pointer ในแฟ้ม

- การควบคุมตำแหน่งของ fp (file pointer) ในแฟ้มข้อมูล นิยมใช้กันมากในการประมวลผลแฟ้มข้อมูลแบบสุ่ม (random file access)
- ซึ่งสามารถให้ fp ไปยังตำแหน่งเริ่มต้นของแฟ้มข้อมูล (BOF =Beginning of File) หรือ ตำแหน่งใดตำแหน่งหนึ่งในแฟ้มข้อมูลได้
- ฟังก์ชันที่ใช้ควบคุมตำแหน่ง file pointer มีดังนี้
- **ฟังก์ชัน rewind( )**
  เป็นฟังก์ชันที่ใช้ย้ายตำแหน่งของ file pointer ไปยังตำแหน่งเริ่มต้นของแฟ้ม
- รูปแบบการใช้ rewind(fp);

ฟังก์ชันที่ใช้ควบคุมตำแหน่งของ file pointer ในแฟ้ม (2)

- **ฟังก์ชัน `fseek( )`**
  เป็นฟังก์ชันที่ใช้ย้ายตำแหน่งของ file pointer ไปยังตำแหน่งที่ต้องการในแฟ้มข้อมูลโดยจะต้องกำหนดจุดเริ่มต้น (origin) ของ file pointer และค่า offset
- **รูปแบบการใช้** `fseek(fp, offset, origin);`
- โดยที่ **offset** คือ ระยะห่างจากตำแหน่งจุดเริ่มต้น มีหน่วยเป็น byte
  origin คือ จุดที่ file pointer ชี้อยู่ มีอยู่ 3 สถานะ ดังนี้คือ
  - SEEK_SET 0 file pointer อยู่ที่ต้นแฟ้ม BOF
  - SEEK_CUR 1 file pointer อยู่ที่ตำแหน่งปัจจุบัน
  - SEEK_END 2 file pointer อยู่ที่ท้ายไฟล์ EOF

ฟังก์ชันที่ใช้ควบคุมตำแหน่งของ file pointer ในแฟ้ม (3)

- ตัวอย่าง แสดงการใช้งาน `fseek(fptr,10,0);`
- หรือคำสั่ง `fseek(fptr,10, SEEK_SET);`
  หมายความว่าให้ย้าย `file pointer` ถัดจากตำแหน่งต้นไฟล์ `(BOF)` ไปอีก 10 `bytes`
- ข้อควรจำ ฟังก์ชัน `fseek( )` จะให้ค่าไม่เท่ากับศูนย์ เมื่อไม่สามารถย้าย `file pointer` ได้

ฟังก์ชันที่ใช้ควบคุมตำแหน่งของ file pointer ในแฟ้ม (4)

- **ฟังก์ชัน `ftell( )`**
  เป็นฟังก์ชันที่ใช้บอกตำแหน่งของ `file pointer` ว่าปัจจุบันกำลังชี้อยู่ที่ตำแหน่งใดในแฟ้มข้อมูล โดยฟังก์ชันนี้จะให้ค่ากลับเป็นตัวเลขจำนวนเต็ม
- รูปแบบการใช้ `ftell( fp );`
- ตัวอย่าง แสดงการใช้งาน `int position;`

```
position = ftell(fp);
printf("Position is %d Byte",position);
```

ฟังก์ชันที่ใช้ควบคุมตำแหน่งของ file pointer ในแฟ้ม (5)

```c
#include <stdio.h>
void main(){
  long loc;
  FILE *Ptr;
  if ((Ptr=fopen("binary.txt","rb"))==NULL){
     printf("Error opening file!");
     return;
  }
  printf("Enter byte to seek:");
  scanf("%d",&loc);
  if (fseek(Ptr, loc, SEEK_SET)){
     printf("Error on seeking file!");
     return;
  }
  printf("data at location %ld is %c", loc, getc(Ptr));
  fclose(Ptr);
}
```