แฟ้มข้อมูล File

แฟ้มข้อมูล (File)

กลุ่มของระเบียนข้อมูล(record) ที่เกี่ยวข้องกับสิ่งเดียวกัน เช่น แฟ้มข้อมูลการลงทะเบียนของนักศึกษา เป็นที่รวมของรายละเอียดการลงทะเบียนของนักศึกษาแต่ละคน ซึ่งเรียกว่าระเบียน

เนื้อหา

- ประเภทของแฟ้มข้อมูล
- ชนิดของแฟ้มข้อมูล
- การประมวลผลด้วยแฟ้มข้อมูล
- การใช้แฟ้มข้อมูลในภาษาซี
- `fopen(), fclose(), feof()`
- `fscanf(), fprintf()`
- `fgetc(), fputc()`
- `fgets(), fpus()`
- `rewind()`
- `fseek(), fread(), fwrite()`

ประเภทของแฟ้มข้อมูล

- แฟ้มข้อมูลจัดแบ่งเป็น 2 ประเภท ตามลักษณะการเข้าถึงข้อมูล (Access mode)
  - แบบลำดับ (Sequential File)
  - แบบสุ่มหรือโดยตรง (Random Access File)
- Sequential file เป็นแฟ้มซึ่งเก็บข้อมูลตามลำดับ(ก่อน-หลัง)ของการบันทึกหรือเขียนลงแฟ้ม ลำดับของการอ่านข้อมูลจะเป็นลำดับเดียวกับการบันทึกข้อมูลลงแฟ้ม
- Random access file เป็นแฟ้มที่เราสามารถเข้าถึงข้อมูลตำแหน่งใดๆในแฟ้มได้ โดยไม่จำเป็นต้องเป็นลำดับเดียวกับการเขียนข้อมูลลงแฟ้ม

ชนิดของแฟ้มข้อมูล

- แบ่งตามลักษณะข้อมูลที่จัดเก็บในแฟ้ม
  - Text file : ข้อมูลถูกจัดเก็บในลักษณะของรหัส ASCII จะสามารถอ่านข้อมูลในแฟ้มได้ เสมือนว่าข้อมูลถูกจัดเก็บเป็นตัวอักษรเรียงต่อเนื่องกันไป
  - Binary file : ข้อมูลถูกจัดเก็บในลักษณะของรหัสเลขฐานสอง จะไม่สามารถอ่านข้อมูลในแฟ้มได้โดยตรง ตัวอย่างของแฟ้มชนิดนี้ ได้แก่ แฟ้มโปรแกรมที่มีสกุลของไฟล์เป็น EXE เป็นต้น

การประมวลผลด้วยแฟ้มข้อมูล

การประมวลผลแฟ้มข้อมูล จำเป็นต้องมีขั้นตอนการทำงานหลัก 4 ขั้นตอน ดังนี้

- **การประกาศตัวแปรพอยน์เตอร์ของแฟ้ม** (File pointer) เพื่อใช้อ้างถึงไฟล์ที่ต้องการดำเนินการ
- **การเปิดแฟ้ม** เพื่อบอกแก่ระบบปฏิบัติการให้รู้ว่ากำลังจะจัดการกับข้อมูลในแฟ้มใด ระบบปฏิบัติการจะจัดเตรียมหน่วยความจำหรือทรัพยากรอื่นที่จำเป็นเพื่อใช้กับแฟ้มที่ขอเปิด
- **การจัดการกับข้อมูลในแฟ้ม** ได้แก่ การอ่าน, การเขียนหรือบันทึก, การกำหนดตำแหน่งข้อมูลเพื่อการอ่านหรือเขียน เป็นต้น
- **การปิดแฟ้ม** เพื่อบอกแก่ระบบปฏิบัติการให้รู้ว่า จะไม่มีการจัดการกับข้อมูลในแฟ้มนั้นๆอีก ระบบปฏิบัติการจะยึดทรัพยากรต่างๆที่จัดสรรให้สำหรับการจัดการแฟ้มนั้นคืน

การใช้แฟ้มข้อมูลในภาษาซี

- ในภาษาซี กำหนดให้มีชนิดข้อมูลพิเศษ เพื่อบ่งชี้ว่า เป็นแฟ้มข้อมูล การกำหนดชนิดข้อมูล ชนิดนี้ใช้คีย์เวิร์ด FILE
- แฟ้มทุกแฟ้มที่จะถูกใช้ในโปรแกรมต้องมีตัวบ่งชี้เฉพาะ(Identifier)ของแต่ละแฟ้ม ตัวบ่งชี้นี้ต้องถูกกำหนดให้มีชนิดข้อมูลเป็นพอยน์เตอร์แบบ FILE
- ประกาศตัวแปรพอยน์เตอร์แบบ FILE ตามรูปแบบต่อไปนี้

FILE * <ชื่อตัวแปร> ;

เช่น FILE * fp ;

หมายความว่า ประกาศให้ ตัวแปร fp เป็นตัวแปรพอยน์เตอร์ ที่เก็บค่าพอยน์เตอร์(address)ของแฟ้มข้อมูล ค่าของตัวแปร fp จะถูกกำหนดด้วยฟังก์ชัน ที่ทำหน้าที่เปิดแฟ้มข้อมูล

การใช้ฟังก์ชัน fopen( )

- ***fopen( )*** เพื่อเปิดแฟ้มข้อมูล
- Include file : <stdio.h>
- Prototype : FILE * fopen (const char * s1, const char * s2);
- Arguments : s1 เป็นชื่อแฟ้มข้อมูลที่ต้องการเปิด,
  s2 เป็นโหมดของการเปิดแฟ้ม
  - "w" : สร้างแฟ้มใหม่เพื่อเขียนหรือบันทึกข้อมูลโดยเริ่มเขียนที่ต้นแฟ้ม
  - "r" : เปิดแฟ้มที่มีอยู่แล้ว เพื่ออ่านข้อมูลโดยเริ่มอ่านจากต้นแฟ้ม
  - "a" : เปิดแฟ้มที่มีอยู่แล้วเพื่อเขียนหรือบันทึกข้อมูลโดยเริ่มเขียนต่อท้ายแฟ้ม หากแฟ้มที่ระบุยังไม่เคยมีมาก่อน แฟ้มนี้จะถูกสร้างขึ้นมาใหม่
  - "w+" : สร้างแฟ้มใหม่เพื่อเขียนและอ่านข้อมูลโดยเริ่มเขียน/อ่านที่ต้นแฟ้ม
  - "r+" : เปิดแฟ้มที่มีอยู่แล้ว เพื่ออ่านและเขียนข้อมูลโดยเริ่มอ่านจากต้นแฟ้ม
  - "a+" : เปิดแฟ้มที่มีอยู่แล้วเพื่อเขียนและอ่านข้อมูลโดยเริ่มเขียน/อ่านที่ท้ายแฟ้ม
- Returns : A pointer to the open file specified by s1 if successful,
  a NULL pointer if unsuccessful

ตัวอย่างการใช้ฟังก์ชัน fopen( )

```
FILE  *fp;    /* ประกาศตัวแปรใช้สำหรับระบุแฟ้ม*/

fp =  fopen("TEST.DAT", "w");
    /* เปิดแฟ้มชื่อ TEST.DAT เพื่อบันทึกข้อมูล
        หากไม่มีแฟ้มนี้อยู่ก่อน จะสร้างขึ้นมาใหม่
        หากมีแฟ้มนี้อยู่แล้ว ข้อมูลเดิมจะถูกลบทิ้งหมด
        ตัวชี้ตำแหน่งข้อมูลในแฟ้มจะอยู่ที่ต้นแฟ้ม */
fp =  fopen("TEST.DAT", "a");
    /* เปิดแฟ้มชื่อ TEST.DAT เพื่อบันทึกข้อมูลต่อท้ายข้อ
        มูลเดิมที่มี
        หากไม่มีแฟ้มนี้อยู่ก่อน จะสร้างขึ้นมาใหม่
        ตัวชี้ตำแหน่งข้อมูลในแฟ้มจะอยู่ที่ท้ายแฟ้ม */
```


การใช้ฟังก์ชัน fclose( )

- ***fclose( )*** เพื่อปิดแฟ้มข้อมูล ที่ได้เปิดใช้
- Include file : <stdio.h>
- Prototype : **fclose** (FILE *fp);
- Arguments : fp เป็นพอยน์เตอร์ของไฟล์ที่ต้องการปิด
  การปิดแฟ้มจะทำสำเร็จ หากแฟ้มนั้นได้เปิดไว้แล้ว
- ตัวอย่าง

```
if (fp = fopen ("example1.txt", "r+")) != NULL)
{
    ...
    fclose(fp);
} else
    printf ("\aFile not found.....\a");
```


การใช้ฟังก์ชัน feof( )

- ***feof( )*** เพื่อตรวจสอบว่าอ่านข้อมูลได้รหัสแสดงการจบแฟ้มหรือไม่
- Include file : <stdio.h>
- Prototype : int **feof** (FILE *fp);
- Arguments : fp เป็นพอยน์เตอร์ของไฟล์ที่ต้องการตรวจสอบ
- Return : ค่าจริง ถ้าหากอ่านได้รหัสการจบแฟ้ม
  ค่าเท็จ หากยังสามารถอ่านข้อมูลได้
- ตัวอย่าง

```
while (!feof(fp)) {
    ...
} ;
```


การใช้ฟังก์ชัน fprintf( )

- ***fprintf( )*** เพื่อเขียนข้อมูลลงแฟ้มตามรูปแบบที่กำหนด
- Include file : <stdio.h>
- รูปแบบการเรียกใช้ : **fprintf** (*file pointer, control string, arg1,...argn*);
- Arguments :
  - *file pointer* เป็นพอยน์เตอร์ของไฟล์ที่ต้องการเขียนข้อมูล
  - control string เป็นรูปแบบของข้อมูลที่ต้องการเขียน
  - *arg1,...argn* เป็นรายการข้อมูลที่ต้องการเขียนลงแฟ้ม โดย arg แต่ละตัวอาจอยู่ในรูปค่าคงที่ / ตัวแปร / นิพจน์
- ตัวอย่าง

```
do {
    printf("Enter your sex(F/M) : "); scanf("%c", &sex);
    printf("                 age : "); scanf("%d", &age);
    fprintf(fp, "%c%d", sex, age);
    printf("To continue press [y/Y]..");
} while (scanf("%[y/Y]", &cont));
```

การใช้ฟังก์ชัน fscanf( )

- ***fscanf( )*** เพื่ออ่านข้อมูลจากแฟ้มตามรูปแบบที่กำหนด
- Include file : <stdio.h>
- รูปแบบการเรียกใช้ : **fscanf (** *file pointer, control string, arg1,...argn* **);**
- Arguments :
  - *file pointer* เป็นพอยน์เตอร์ของไฟล์ที่ต้องการอ่านข้อมูล
  - control string เป็นรูปแบบของข้อมูลที่ต้องการอ่าน
  - *arg1,...argn* เป็น address ของตัวแปรที่ใช้รับค่าข้อมูลที่ต้องการอ่านจากแฟ้ม
- ตัวอย่าง

```
fscanf(fp, "%c%d", &sex, &age);
while (!feof(fp)) {
    printf("%c\t%2d", sex, age);
    fscanf(fp, "%c%d", &sex, &age);
}
fclose();
```


การใช้ฟังก์ชัน rewind( )

- ***rewind( )*** เพื่อเซ็ตให้ตัวชี้ตำแหน่งข้อมูลในแฟ้ม ย้อนกลับมาอยู่ที่ต้นแฟ้ม
- Include file : <stdio.h>
- รูปแบบการเรียกใช้ : **rewind (*file pointer*);**
- Arguments : *file pointer* เป็นพอยน์เตอร์ของไฟล์ที่ต้องการกำหนดให้ตัวชี้ตำแหน่งข้อมูลในแฟ้ม
- ตัวอย่าง

```
fp = fopen("myword.txt", "w+");
do {
    printf("Enter your words : ");
    scanf("%s", text);
    fprintf(fp, "%s", text);
    printf("To continue press [y/Y]..");
} while (scanf("%[y/Y]", &cont));
rewind(fp);
while (fscanf(fp, "%s", text))
printf("%s\n", text);
fclose();
```


แบบฝึกหัด

- จงเขียนโปรแกรมรับชื่อเกมส์ที่คุณชอบ แล้วบันทึกลงไฟล์

การอ่านและพิมพ์ทีละบรรทัด

- ***fgets() และ fputs()*** เป็นฟังก์ชันที่ทำหน้าที่ในการอ่านและเขียนข้อมูลทีละบรรทัด ตามลำดับ
- Include file : <stdio.h>
- รูปแบบการเรียกใช้ : `fgets(buffer, n, file_pointer );`
  `fputs(buffer, file_pointer);`
- Arguments : *buffer* คือพอยน์เตอร์ที่ชี้ไปยังชุดของอักขระที่ต้องการอ่านหรือเขียนข้อมูล
  n เป็นความยาวหรือขนาดของ buffer
  file pointer เป็นพอยน์เตอร์ของไฟล์ที่ต้องการกำหนดให้ตัวชี้ตำแหน่งข้อมูลในแฟ้ม
- ในการอ่านนั้น fgets() จะทำการอ่านจนกว่าจะเจออักขระ '\n' หรือจนกว่าจะอ่านได้ครบ n-1 ตัวอักษร ซึ่งกรณีนี้จะใส่ null character เป็นอักขระสุดท้ายใน buffer
- ในการเขียน fputs() จะนำข้อมูลจาก buffer ไปเขียนลงในไฟล์ที่ระบุจนกว่าจะเจออักขระ null แล้วจะเขียน '\n' ลงในไฟล์แทนอักขระ null

ตัวอย่างการอ่านและพิมพ์ทีละบรรทัด

```
/* copy content of "source"  into "destination" file*/
#define Max 256
int filecopy(char *source, char *destination) {
    FILE  *fs, *fd;
    char  buff[Max];
    if ((fs = fopen(source, "r")) == NULL) {
        printf("%s not found\n\a\a", source);
        return (-1);
    }
    if ((fd = fopen(destination, "w")) == NULL) {
        printf("%s not found\n\a\a", destination);
        return (-2);
    }

    while (fgets(buff, sizeof(buff), fs))
        fputs(buff, fd);
    fclose(fs); fclose(fd);
}
```


การอ่านและพิมพ์ทีละบรรทัด

- ***fgets() และ fputs()*** เป็นฟังก์ชันที่ทำหน้าที่ในการอ่านและเขียนข้อมูลทีละบรรทัด ตามลำดับ
- Include file : <stdio.h>
- รูปแบบการเรียกใช้ : `fgets(buffer, n, file_pointer );`
  `fputs(buffer, file_pointer);`
- Arguments : *buffer* คือพอยน์เตอร์ที่ชี้ไปยังชุดของอักขระที่ต้องการอ่านหรือเขียนข้อมูล
  n เป็นความยาวหรือขนาดของ buffer
  file pointer เป็นพอยน์เตอร์ของไฟล์ที่ต้องการกำหนดให้ตัวชี้ตำแหน่งข้อมูลในแฟ้ม
- ในการอ่านนั้น fgets() จะทำการอ่านจนกว่าจะเจออักขระ '\n' หรือจนกว่าจะอ่านได้ครบ n-1 ตัวอักษร ซึ่งกรณีนี้จะใส่ null character เป็นอักขระสุดท้ายใน buffer
- ในการเขียน fputs() จะนำข้อมูลจาก buffer ไปเขียนลงในไฟล์ที่ระบุจนกว่าจะเจออักขระ null แล้วจะเขียน '\n' ลงในไฟล์แทนอักขระ null

แบบฝึกหัด

- จงเขียนโปรแกรม เพื่อทำหน้าที่ คัดลอกไฟล์ โดยมีรูปแบบการเรียกใช้โปรแกรม ดังนี้
- mycopy <ชื่อไฟล์ต้นฉบับ> <ชื่อไฟล์สำเนา>