การดำเนินการกับแฟ้มข้อมูล (File)

310-230 การโปรแกรมเชิงโครงสร้างและการประยุกต์

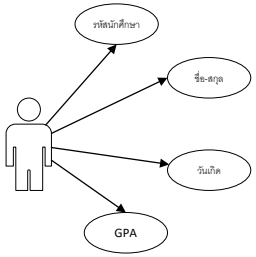
แฟ้มข้อมูล (File)

เป็นการจัดการข้อมูลบนหน่วยความจำสำรอง (secondary storage) หรือสื่อบันทึกข้อมูล

โครงสร้างของแฟ้มข้อมูลประกอบด้วย

- ระเบียนข้อมูล (record) ประกอบขึ้นมาจากรายการข้อมูล (data item) เรียกว่า field ซึ่งสามารถประกอบด้วยรายการข้อมูลย่อยๆได้อีก

ตัวอย่างโครงสร้างแฟ้มข้อมูลและระเบียนข้อมูล



1000, Jane Foster, 25/03/1990
1001, Billy James, 17/06/1991
1002, Lars Aldrich, 11/01/1990

- อาร์เรย์ structure และ union เป็นรูปแบบการจัดการข้อมูลในหน่วยความจำหลัก
- แฟ้มข้อมูลเป็นรูปแบบการจัดการข้อมูลในหน่วยความจำสำรอง

การเข้าถึงข้อมูลในแฟ้มข้อมูล

การเข้าถึงแบบตามลำดับ

- ต้องผ่านข้อมูลที่อยู่ใน record ก่อนหน้าตามลำดับเพื่อไป record ที่ต้องการ
- ใช้กับแฟ้มข้อมูลที่บันทึกบนสื่อบันทึกข้อมูลเช่น เทปแม่เหล็ก จานแม่เหล็ก

การเข้าถึงแบบโดยตรง

- สามารถไปยัง record ที่ต้องการโดยตรง
- ใช้กับแฟ้มข้อมูลที่บันทึกบนสื่อบันทึกข้อมูลเช่น จานแม่เหล็ก

การจัดการแฟ้มข้อมูล หรือการจัดโครงสร้างแฟ้มข้อมูล (File organization)

เทคนิคในการจัดเก็บจะมีผลต่อการใช้งานแฟ้มข้อมูลภายหลังเช่น

- การเข้าถึงข้อมูล
- การจัดเก็บข้อมูลเพิ่มเติม
- การอ่านข้อมูลเป็นต้น

การจัดการแฟ้มข้อมูล หรือการจัดโครงสร้างแฟ้มข้อมูล (File organization)

เทคนิคการจัดโครงสร้างแฟ้มข้อมูล

- ไม่มีโครงสร้างช่วยในการเข้าถึงข้อมูลในแฟ้มข้อมูลเช่น sequential file เป็นต้น
- ใช้ดัชนีช่วยในการเข้าถึงข้อมูลในแฟ้มข้อมูลเช่น indexed-sequential file เป็นต้น
- ใช้วิธีการของคอมพิวเตอร์ช่วยในการเข้าถึงข้อมูลเช่น direct file เป็นต้น

การดำเนินการกับแฟ้มข้อมูล

การเปิดแฟ้มข้อมูล

การปิดแฟ้มข้อมูล

การตรวจสอบจุดสิ้นสุดของแฟ้มข้อมูล

การอ่านแฟ้มข้อมูล

การเขียนแฟ้มข้อมูล

การเปิดแฟ้มข้อมูล

สามารถอ้างถึง หรือเข้าถึงแฟ้มข้อมูลด้วยตัวแปรชนิดพอยน์เตอร์

ตัวแปรพอยน์เตอร์ที่ใช้เรียกว่า ไฟล์พอนต์เตอร์ (File pointer)

ดังอย่างเช่น

FILE *fp1, *fp2;

fp1 เป็นพอยน์เตอร์ชี้ไปยังแฟ้มข้อมูลที่ 1

fp2 เป็นพอยน์เตอร์ชี้ไปยังแฟ้มข้อมูลที่ 2

ระบุ header file ด้วย stdio.h

การเปิดแฟ้มข้อมูล

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

ฟังก์ชัน **fopen** ส่งค่ากลับเป็นตำแหน่งที่อยู่ของแฟ้มข้อมูล

- ถ้าการเปิดแฟ้มข้อมูลไม่มีข้อผิดพลาด ฟังก์ชัน fopen จะส่งค่า ***ตำแหน่งที่อยู่*** ให้กับตัวแปรพอยน์เตอร์ที่ประกาศไว้
- ถ้าเกิดข้อผิดพลาดในการเปิดแฟ้มข้อมูล ฟังก์ชัน fopen จะส่งค่ากลับเป็น ***null pointer*** หรือ ***NULL***

การเปิดแฟ้มข้อมูล

Mode

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

Mode	คำอธิบาย	สร้างไฟล์ใหม่	อ่านข้อมูล	เขียนข้อมูล
r	เปิดไฟล์ที่มีอยู่ก่อนแล้ว	-	ใช่	-
w	เปิดไฟล์เพื่อเขียนทับ	ใช่	-	เขียนทับ
a	เปิดไฟล์เพื่อเขียนเพิ่ม	ใช่	ใช่	เขียนต่อ
r+	เปิดไฟล์เพื่ออ่านและเขียน	-	ใช่	เขียนทับ
w+	เปิดไฟล์เพื่ออ่านและเขียนทับ (ล้างข้อมูลก่อน)	ใช่	ใช่	เขียนทับ
a+	เปิดไฟล์เพื่ออ่านตั้งแต่เริ่มและเขียนต่อท้าย	ใช่	ใช่	เขียนต่อ

การเปิดแฟ้มข้อมูล

การตรวจสอบผลการเปิดไฟล์

```
int main() {
   FILE *fp;
   fp = fopen("data1.dat", "r");

   if (fp == NULL) {
      printf("Cannot open file data1.dat", fp);
   }
   return 0;
}
```


การปิดแฟ้มข้อมูล

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

fclose เป็นฟังก์ชันมาตรฐานของภาษาซีสำหรับปิดแฟ้มข้อมูล มีรูปแบบการเรียกใช้งานดังนี้

fclose(fp);

หมายถึง การปิดแฟ้มข้อมูลที่มีไฟล์พอยน์เตอร์ fp ชี้อยู่

ฟังก์ชัน fclose ส่งค่ากลับเป็น **int**

ถ้าปิดแฟ้มข้อมูลได้สำเร็จจะส่งค่า **0** กลับ

ถ้าไม่สามารถปิดแฟ้มข้อมูลจะส่งค่า **EOF** (**E**nd **o**f **F**ile) ซึ่งจะถูกกำหนดค่าเป็น -1

การปิดแฟ้มข้อมูล

```
int main() {
   FILE *fp;
   int close;
   fp = fopen("data1.dat", "r");
   close = fclose(fp);
   if (close == EOF) { printf("Cannot close file data1.dat"); }
   else { printf("File data1.dat closed"); }
   return 0;
}
```


การตรวจสอบจุดสิ้นสุดของแฟ้มข้อมูล

ฟังก์ชัน **feof** เป็นฟังก์ชันที่ใช้ทดสอบว่าตำแหน่งที่กำลังดำเนินการกับข้อมูลในแฟ้มข้อมูลเป็นจุดสิ้นสุดของแฟ้มข้อมูลหรือไม่

```
int feof( FILE *fp );
```

Parameters

fp คือ file pointer ที่ชี้ไปยังแฟ้มข้อมูลที่ต้องการอ่านข้อมูล

Return Value

0 เมื่อ file pointer ถึงจุดสิ้นสุดของไฟล์แล้ว

อื่น ๆ เมื่อ file pointer ยังไม่ถึงถึงจุดสิ้นสุดของไฟล์

การอ่านแฟ้มข้อมูล

fgetc() อ่านทีละตัวอักษรจนกว่าจะเจอ EOF

fgets() อ่านทีละ 1 บรรทัด (จนกว่าจะเจอ '\n') หรือ EOF

fscanf() อ่านจนกว่าจะเจอช่องว่าง (space)

การอ่านแฟ้มข้อมูล

fgetc()

```
int fgetc( FILE *fp );
```

เป็นฟังก์ชันสำหรับอ่านข้อมูลชนิดตัวอักษรคราวละ 1 ตัวอักษร

รูปแบบการเรียกใช้งาน

c = getc(fp);

- อ่านข้อมูลคราวละ 1 ตัวอักษรจากแฟ้มข้อมูลที่มี ไฟล์พอยนเตอร์ fp ชี้อยู่
- เมื่ออ่านจนจบแฟ้มข้อมูลค่าที่ส่งกลับคือค่า EOF

การอ่านแฟ้มข้อมูล

fgetc() > ตัวอย่าง

```
FILE *fp; int close; char c;
fp = fopen("data2.dat", "r");
printf("#### DATA in data2.dat ####\n");
do {
   c = fgetc(fp);
   if (feof(fp)) { break; }
   printf("%c", c);
} while (1);
fclose(fp);
```

```
#### DATA in data2.dat ####
I love BABYMETAL.
I hate an insect.
```


การอ่านแฟ้มข้อมูล

fgets()

```
char fgets( char *line, int size, FILE *fp );
```

Parameters

line คือ pointer ที่ชี้ไปยังจุดเริ่มต้นของอาร์เรย์ของ buffer สำหรับตัวอักษรที่จะอ่าน

size คือจำนวนตัวอักษรสูงสุดต่อบรรทัด

fp คือ file pointer ที่ชี้ไปยังแฟ้มข้อมูลที่ต้องการอ่านข้อมูล

Return Value

อ่านข้อมูลการอ่านจะอ่านจนกว่าจะพบ '\n' หรืออ่านไปครบ size -1 ตัวอักษร

- ถ้าอ่านข้อมูลได้สำเร็จ line pointer จะถูกส่งกลับ
- ถ้าเกิดข้อผิดพลาดจะส่งค่า null pointer กลับ

การอ่านแฟ้มข้อมูล

fgets() > ตัวอย่าง

```
FILE *fp; int close; char buffer[100];
fp = fopen("data1.dat", "r");
...
do {
   if (fgets(buffer, 100, fp) != NULL) {
      printf("LINE: %s\n", buffer);
   } else { break; }
} while (1);
```

```
#### DATA in data1.dat ####
LINE: 1000, Jane Foster, 25/03/1990

LINE: 1001, Billy James, 17/06/1991

LINE: 1002, Lars Aldrich, 11/01/1990


File data1.dat closed
```


การอ่านแฟ้มข้อมูล

fscanf()

```
int fscanf( FILE *fp, const char  *format, … );
```

Parameters

fp คือ file pointer ที่ชี้ไปยังแฟ้มข้อมูลที่ต้องการอ่านข้อมูล

format คือรูปแบบการรับข้อมูลเข้า (เหมือน scanf)

การอ่านแฟ้มข้อมูล

fscanf() > ตัวอย่าง

```
FILE *fp; int close, age;
char word1[30], word2[30];
fp = fopen("data3.dat", "r");
…
fscanf(fp, "%s %s %d", word1, word2, &age);
printf("RESULT: %s | %s | %d", word1, word2, age);
```

```
RESULT: I | am | 25
File data3.dat closed
```


การเขียนแฟ้มข้อมูล

fputc() เขียนทีละตัวอักษรไปยัง stream

puts() เขียนทีละ 1 สายอักขระ (ไม่เพิ่ม '\n' และ null ให้ ต้องเพิ่มเอง) ไปยัง stream

fprintf() เขียนอักขระไปยัง stream ตาม format

การอ่านแฟ้มข้อมูล

fputc()

```
int fputc( int ch, FILE *fp );
```

Parameters

ch คือ ตัวอักษรที่ต้องการเขียน

fp คือ file pointer ที่ชี้ไปยังแฟ้มข้อมูลที่ต้องการเขียนข้อมูล

การอ่านแฟ้มข้อมูล

fputc() > ตัวอย่าง

```
FILE *fp; int ch, close;
fp = fopen("output_ex6.dat", "w");
…
for(ch = 33; ch <= 100; ch++) {
   fputc(ch, fp);
}

close = fclose(fp);
```

```
!"#$%&'()*+,-./0123456789:;<=>?@
```

การอ่านแฟ้มข้อมูล

fputs()

```
int fputs( char *line, FILE *fp );
```

Parameters

line คือ pointer ที่ชี้ไปยังจุดเริ่มต้นของ string ที่จะเขียน

fp คือ file pointer ที่ชี้ไปยังแฟ้มข้อมูลที่ต้องการเขียนข้อมูล

Return Value

- Non-negative value ถ้าเขียนข้อมูลได้สำเร็จ line pointer จะถูกส่งกลับ
- EOF ถ้าเกิดข้อผิดพลาด

การอ่านแฟ้มข้อมูล

fputs() > ตัวอย่าง

```
FILE *fp; int ch, close;
char name[100] = "I am John Hutton.";
fp = fopen("output_ex7.dat", "w");
…
fputs("This is the first line.", fp);
fputs(name, fp);
fputs("End!!!", fp);
…
```

```
This is the first line.I am John Hutton.End!!!
```


การอ่านแฟ้มข้อมูล

fprintf()

```
int fprintf( FILE *fp, const char  *format, … );
```

Parameters

fp คือ file pointer ที่ชี้ไปยังแฟ้มข้อมูลที่ต้องการอ่านข้อมูล

format คือรูปแบบการเขียนข้อมูล (เหมือน printf)

การอ่านแฟ้มข้อมูล

fprintf() > ตัวอย่าง

```
FILE *fp;
int ch, close, age = 36;
float weight = 85.2358;
char name[100] = "John Cooper";
…
fprintf(fp, "%s %d %.2f", name, age, weight);
```

```
John Cooper 36 85.24
```