บทที่ 13 การจัดการไฟล์ (File Management)

รายวิชา สธ 113 การออกแบบโปรแกรมทางธุรกิจเบื้องต้น

อ.อภิพงศ์ ปิงยศ

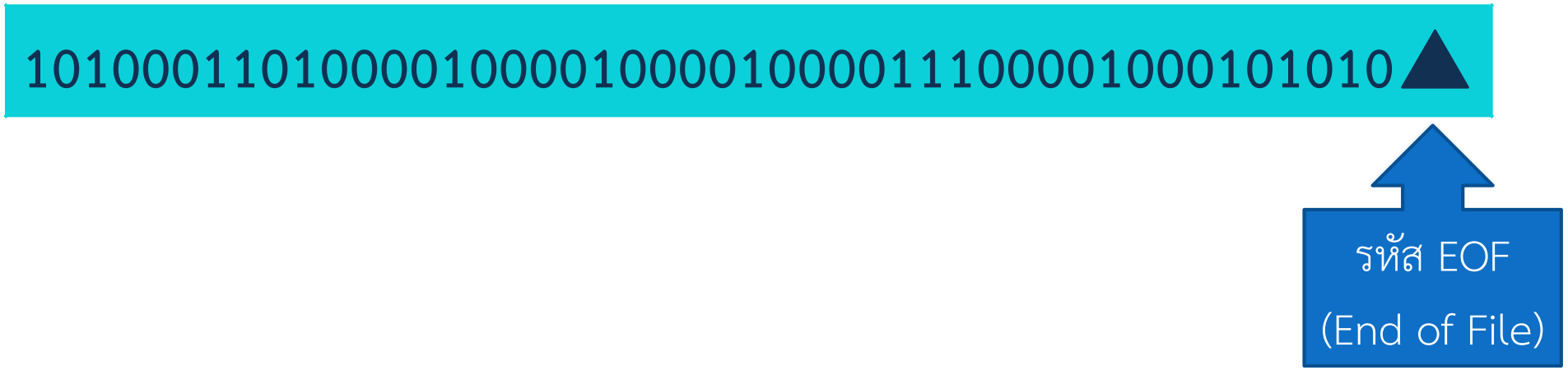
Overview

- การจัดการไฟล์
- ประเภทของไฟล์
- พื้นฐานการทำงานกับไฟล์
- การอ่านและเขียนเท็กซ์ไฟล์

การจัดการไฟล์

- งานบางประเภทต้องมีการนำผลการทำงานของโปรแกรมเก็บลงในหน่วยความจำสำรองในรูปแบบไฟล์ เพื่อสามารถนำมาใช้ในภายหลังได้
- **สตรีม (Stream)** เป็นหน่วยของข้อมูลที่เรียงติดกัน ไฟล์จะติดต่อกับสตรีมเพื่อรับ-ส่งข้อมูล แบ่งเป็น 2 ประเภท คือ
  - **Text Stream** จะเก็บข้อมูลเป็นรหัส ASCII
  - **Binary Stream** จะเก็บข้อมูลในรูปแบบเลขฐานสอง

ลักษณะของ Stream ที่มีการเก็บข้อมูลแบบเรียงต่อกัน



ประเภทของไฟล์

มี 2 ประเภท คือ

- ไฟล์ตัวอักษร (Text File)
- ไฟล์ข้อมูล (Binary File)

Text File

- เป็นไฟล์ข้อความ ข้อมูลจะเก็บเป็นรหัส ASCII
- มีรหัสควบคุมใช้ในการขึ้นบรรทัดใหม่ หรือการจบไฟล์ เป็นต้น
- สามารถใช้โปรแกรม Editor หรือ Notepad ในการเปิดไฟล์ประเภทนี้ได้

ตัวอย่างของข้อมูลในรูปแบบ Stream และ Text File

A	B	¬	2	¬	
01000001	01000010	00001101	00110010	00001101	▲

Stream

Text

Binary File

- เป็นไฟล์ที่เก็บข้อมูลเลขฐานสองของข้อมูลจริงๆ เช่น เลข 2 ในรหัสแอสกีจะเก็บเป็น 00110010 แต่ไฟล์ไบนารีจะเก็บค่าเลขฐานสองที่มีค่าเท่ากับสองจริงๆ (00000010)
- ไม่สามารถใช้โปรแกรมเปิดไฟล์ประเภทนี้มาอ่านได้

การเก็บค่าแบบ Binary

- ประกาศ int x = 16706
- แปลงเป็นเลขฐานสองขนาด 2 Bytes ได้เป็น

01000001	01000010	

พื้นฐานการทำงานกับไฟล์

- ในการเขียนหรืออ่านไฟล์ จะต้องใช้ตัวแปรที่เรียกว่า “File Pointer”
- File Pointer จะเป็นตัวบอกว่าตำแหน่งที่กำลังดำเนินการอยู่ชี้อยู่ที่ตำแหน่งใดของไฟล์
- หากไม่มี File Pointer จะไม่สามารถกระทำการใดๆกับไฟล์ได้เลย

กระบวนการกระทำกับไฟล์

- 1) **เปิดไฟล์** เพื่อให้ระบบรู้ว่าต้องการติดต่อกับไฟล์ใด
- 2) **กระทำการกับไฟล์** ทำการอ่านหรือเขียนข้อมูล
- 3) **ปิดไฟล์** เป็นการบอกว่ากระทำการกับไฟล์นั้นเสร็จเรียบร้อยแล้ว

การเปิดไฟล์

- จะใช้ฟังก์ชัน fopen() ที่อยู่ในไลบรารี stdio.h
- มีรูปแบบคือ
  - File *fp;
  - fp = fopen(filename, mode)

Mode ในฟังก์ชัน fopen

Mode	การทำงาน
r	**เปิดเท็กซ์ไฟล์เพื่ออ่าน**
w	**สร้างเท็กซ์ไฟล์ใหม่เพื่อเขียน**
a	**เปิดเท็กซ์ไฟล์เพื่อเขียนข้อมูลต่อท้าย**
rb	เปิดไบนารีไฟล์เพื่ออ่าน
wb	สร้างไบนารีไฟล์เพื่อเขียน
ab	สร้างไบนารีไฟล์เพื่อเขียนข้อมูลต่อท้าย
r+	**เปิดเท็กซ์ไฟล์เพื่ออ่านหรือเขียนทับไฟล์เก่า**
w+	เปิดเท็กซ์ไฟล์เพื่ออ่านหรือเขียนทับไฟล์เก่า หรือไฟล์ใหม่
a+	เปิดเท็กซ์ไฟล์เพื่ออ่านหรือเขียนต่อท้ายไฟล์เก่า หรือเขียนไฟล์ใหม่
r+b	เปิดไบนารีไฟล์เพื่ออ่านหรือเขียนทับไฟล์เก่า
w+b	เปิดไบนารีไฟล์เพื่ออ่านหรือเขียนทับไฟล์เก่า หรือไฟล์ใหม่
a+b	เปิดไบนารีไฟล์เพื่ออ่านหรือเขียนทับไฟล์เก่า หรือเขียนไฟล์ใหม่

การเขียนไฟล์ใหม่

- File *fp;
- fp = fopen(“d:\\data.txt”, “w”);

- เป็นการเขียนไฟล์ใหม่ชื่อว่า data.txt โดยจะจัดเก็บไฟล์เอาไว้ในไดร์ฟ D

การเปิดไฟล์เพื่ออ่านข้อมูล

- การเปิดอ่านไฟล์จะต้องมีการตรวจสอบก่อนว่ามีไฟล์นั้นอยู่จริงหรือไม่ ถ้าไม่มีไฟล์หรือไม่สามารถเปิดได้จะมีการส่งค่า NULL ออกมา
- ตัวอย่างการตรวจสอบการเปิดไฟล์ เช่น

```
FILE *fp;
if (fp = fopen("D:\\myfile.txt", "r") == NULL)
{
        printf("Error opening file\n");
        exit(1);
}
```

การปิดไฟล์

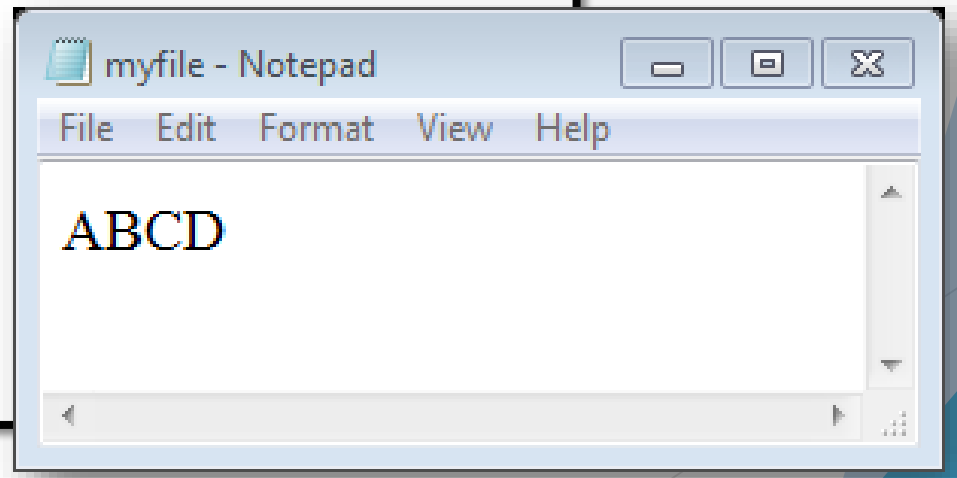
- จะใช้ฟังก์ชัน fclose() ในการปิดไฟล์
- มีรูปแบบคือ
  - **fclose(fp);**
  - เมื่อ fp คือตัวแปรไฟล์พอยเตอร์

การดำเนินการกับไฟล์

- ฟังก์ชันสำหรับอ่านเขียนข้อมูลครั้งละไบต์ จะใช้ฟังก์ชัน
- **fgetc(fp)** จะทำงานคล้ายกับ getchar() แต่จะอ่านข้อมูลออกมาหนึ่งไบต์จากตำแหน่งที่ไฟล์พอยเตอร์ชี้อยู่
- fputc(ch, fp) จะทำงานคล้ายกับฟังก์ชัน putchar() แต่จะเขียนข้อมูลหนึ่งไบต์ที่กำหนดในตัวแปร ch ลงไปในไฟล์ในตำแหน่งที่พอยเตอร์ชี้อยู่

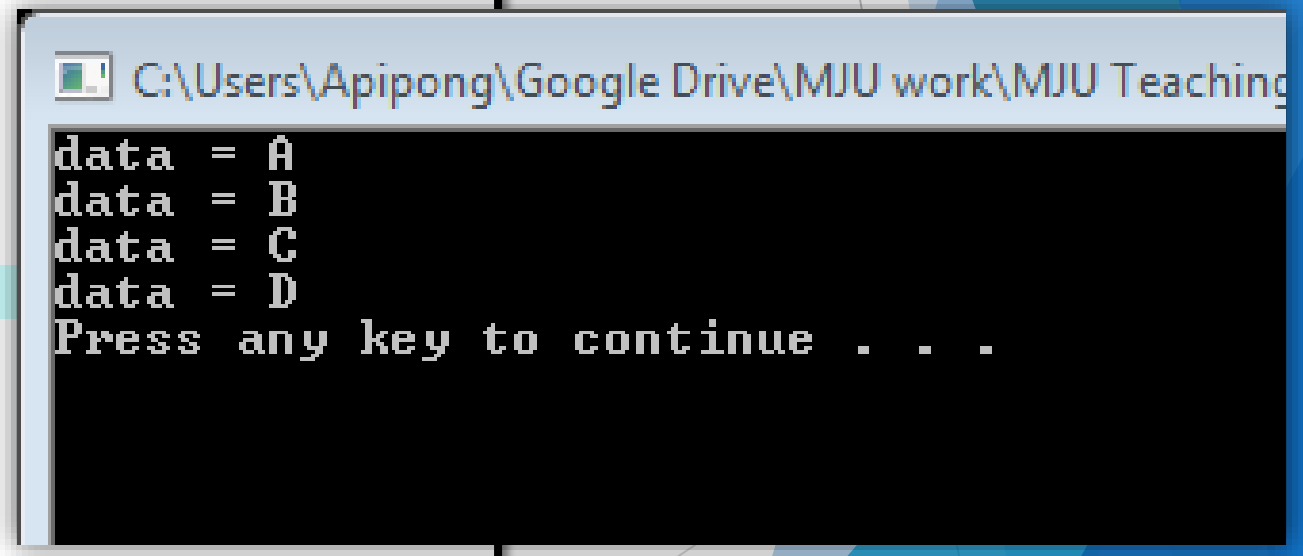
ตัวอย่างที่ 1 การเขียนตัวอักษรลงไฟล์จำนวน 4 ตัวอักษร

```c
#include "stdio.h"
#include "stdlib.h"
int main()
{
        FILE *fp;
        if((fp = fopen("d:\\myfile.txt", "w"))==NULL)
        {
                printf("Cannot open file\n");
                exit(1);
        }
        fputc('A', fp);
        fputc('B', fp);
        fputc('C', fp);
        fputc('D', fp);
        fclose(fp);
        return 0;
}
```



ตัวอย่างที่ 2 อ่านไฟล์ที่ได้จากการเขียนด้วยตัวอย่างที่ 1 มาแสดงผล

```c
{
    char c;
    FILE *fp;
    if((fp = fopen("d:\\myfile.txt", "r"))==NULL)
    {
        printf("Cannot open file\n");
        exit(1);
    }
    c = fgetc(fp);
    printf("data = %c\n", c);
    c = fgetc(fp);
    printf("data = %c\n", c);
    c = fgetc(fp);
    printf("data = %c\n", c);
    c = fgetc(fp);
    printf("data = %c\n", c);
    fclose(fp);
    system("PAUSE");
    return 0;
}
```



การวนลูปอ่านตัวอักษรทั้งหมดในไฟล์

- หากมี Text File ที่เก็บตัวอักษรไว้เป็นจำนวนมาก และต้องการอ่านขึ้นมาแสดงผลทางจอภาพ จะใช้คำสั่ง fgetc() ร่วมกับลูป while ในการวนลูป และตรวจสอบเงื่อนไขว่าเป็นจุดสิ้นสุดของไฟล์หรือยัง (End of File) ด้วยฟังก์ชัน feof(fp)

```
char c;
while(!feof(fp))
{
      c = fgetc(fp);
      printf("%c", c);
}
```

การอ่านและเขียน Text File

- มีฟังก์ชันที่เกี่ยวกับการอ่าน-เขียนเท็กซ์ไฟล์โดยเฉพาะอยู่ 4 ฟังก์ชัน คือ
  - **fputs(ข้อความ string, fp)** ใช้เขียนสตริงที่อยู่ในตัวแปร
  - ***fgets(ตัวแปร string, length, fp)** ใช้อ่านตัวอักษรมาเก็บไว้ในตัวแปรสตริง โดยจะอ่านตัวอักษรจำนวนเท่ากับ length หรือจนกว่าจะขึ้นบรรทัดใหม่ หรืออ่านจนจบไฟล์จึงจะหยุดอ่าน
  - **fprintf(fp, control_string)** ทำงานเหมือนกับ printf() แต่จะใช้เฉพาะกับไฟล์เท่านั้น
  - **fscanf(fp, control_string)** ทำงานเหมือนกับ scanf() แต่จะใช้เฉพาะกับไฟล์เท่านั้น

ตัวอย่างที่ 3 การใช้ fputs เพื่อเขียนสตริงลงในไฟล์

```c
#include "stdio.h"
#include "stdlib.h"
int main()
{
        FILE *fp;
        if((fp = fopen("d:\\myfile.txt", "w"))==NULL)
        {
                printf("Cannot open file\n");
                exit(1);
        }
        fputs("DATA DATA DATA DATA\n", fp);
        fputs("COMPUTER COMPUTER\n", fp);
        fputs("PROGRAM.........\n", fp);
        fputs("Test Test Test Test\n", fp);
        fclose(fp);
        return 0;
}
```