# Storage and File Structure

Type of Storage

| | |
|---|---|
| 1. Volatile storage | ไฟดับ ข้อมูลหาย |
| 2. Non- volatile storage | ไฟดับ ข้อมูลไม่หาย |

Physical Storage

| | | | |
|---|---|---|---|
| cache | volatile | เร็วสุด แพงสุด | |
| Main Memory | volatile | เร็ว | ex. RAM |
| Flash Memory | non-volatile | อ่านเร็ว, เขียน/ลบ ช้า | ex. Flash drive |
| Magnetic Disk | non-volatile | direct-access | ex. Hard Disk |
| Optical Storage | non-volatile | | ex. CD-ROM |
| Tap Storage | non-volatile | sequential-access | |

Storage Hierarchy

| | | | |
|---|---|---|---|
| 1. Primary storage | volatile | cache, main memory | |
| 2. Secondary storage | non-volatile | Flash memory, Magnetic Disk | (on-line storage) |
| 3. Tertiary storage | non-volatile | Optical storage, Magnetic Tape | (off-line storage) |

**Magnetic Disk**

ส่วนประกอบ

| | |
|---|---|
| platter | แต่ล่ะแผ่นของ disk |
| track | platter แบ่งเป็นวงๆ แต่ล่ะวงเรียก track |
| sector | track แบ่งเป็นส่วนๆเท่าๆกัน แต่ล่ะส่วนเรียก sector |
| Cylinder | sector ต่าง platter แต่อยู่ตรงกัน |
| Read-Write Head | หัวอ่าน จะไม่ติด disk ลอยอยู่เล็กน้อย |
| Arm | แขนของหัวอ่าน เอาไว้เลื่อนเข้า/ออก |

Head-disk Assemblies

Disk controller

- เป็นส่วนเชื่อมต่อระหว่างระบบคอมพิวเตอร์(computer system) และตัวฮาร์ดแวร์(disk drive)
- คอยควบคุม และจัดการการอ่าน/เขียน ของ disk

Disk interface มาตรฐานการเชื่อมต่อของ Disk ex. ATA, SATA, SCSI

Performance Measures of Disks คุณภาพของ disk

- Access time เวลาที่ใช้ในการอ่าน/เขียน แบ่งออกเป็น
  - Seek time เวลาหัวอ่านใช้ในการเลื่อนเข้า/ออก

By Natthapach Anuwattananon

Rotational latency อัตราเร็วในการหมุนของ disk

- Data-transfer rate อัตราเร็วในการส่งข้อมูล
- Mean time to failure (MTTF) อายุการใช้งาน ส่วนมากประมาณ 3-5 ปี

Optimization of Disk-Block Access

Block คือ sector ที่อยู่ติดกันเป็นลำดับภายใน track เดียวกัน
ข้อมูลจะถูกส่งจาก disk ไปยัง main memory เป็น blockๆ

Disk-arm-scheduling คือ algorithm ในการทำงานของ arm

- elevator algorithm ถ้าเข้าอยู่จะเข้าให้หมดก่อนค่อยออก
ถ้าออกอยู่จะออกให้หมดก่อนค่อยเข้า

File organization ลดเวลาในการเข้าถึง block ด้วยการเก็บข้อมูลที่สัมพันธ์กันไว้ใน block ใกล้ๆกัน

- fragment เมื่อทำการแก้ไข/ลบ ไฟล์ จะทำให้เกิด block ว่างๆ กระจายตัวอยู่จำนวนมาก
และเมื่อจะสร้าง file ใหม่ fileใหม่นั้นจะใช้พื้นที่ว่างที่กระจายอยู่
ทำให้เวลาเข้าถึงใช้เวลานาน
- defragment กระบวนการที่ใช้ในการรวบรวม fragment เพื่อลดเวลาในการเข้าถึง

Nonvolatile write buffers

- เนื่องจาก disk เขียนได้ช้า การจะให้มานั่งรอเขียนให้เสร็จเลยเสียเวลามาก
- จึงได้เพิ่ม Non-volatile RAM เข้ามาทำหน้าที่เป็น buffer
- เวลาเขียนจะเขียนลงไปใน buffer ก่อน แล้วค่อยเขียนลง disk

Log disk

- เขียนเร็ว เพราะไม่มี seek time

**RAID : Redundant Arrays of Independent Disks**

- เทคนิคในการจัดการ disk จำนวนมากให้ทำงานได้เหมือน disk เดียว

Improvement of Reliability via Redundancy ปรับปรุงความน่าเชื่อถือด้วย Redundancy

Redundancy การเก็บข้อมูลพิเศษไว้เพื่อใช้ในการสร้างข้อมูลที่เสียหายไปขึ้นมาใหม่ เช่น

Mirroring (shadowing)

- คัดลอก(Duplicate) ทุกๆ disk โดยการใช้ 2 disk เก็บข้อมูล
- เวลาเขียนจะเขียนทั้ง 2 disk, เวลาอ่านจะอ่านจากอันไหนก็ได้
- ถ้า 1ใน2 disk พัง ข้อมูลจะยังคงอยู่ในอีกอันนึง
- ข้อมูลจะหายได้ก็ต่อเมื่อ disk ล่มทั้ง2อันก่อนที่ระบบจะ reapair

Mean time to repair ระยะเวลาเฉลี่ยในการ repair ข้อมูล

Mean time to data loss ระยะเวลาเฉลี่ยนที่ข้อมูลจะหาย โดยจะขึ้นอยู่กับระยะเวลา repair และ MTTF

- ปรับปรุงเวลาในการส่งข้อมูลโดยการ strip ข้อมูลระหว่าง disk

Bit-level striping

- แบ่งข้อมูลแต่ล่ะ byte ออกเป็น 8 bit เก็บไว้ใน 8 disk
- Each access can read data at eight times the rate of a single disk
- แต่ว่า seek/access time จะแย่กว่าอ่านจาก disk เดียว

Block-level striping

- แบ่งข้อมูลใน file ออกเป็น blockๆ เก็บใน disk แต่ล่ะอัน
- เมื่อมี n disk, block ที่ i จะถูกเก็บใน disk ที่ (i%n) +1

RAID Levels

- รูปแบบในการเตรียม redundancy ในราคาต่ำ โดยใช้ data striping ร่วมกับ parity bits
- RAID level ที่ต่างกันจะมีค่าใช้จ่ายต่างกัน วิธีการต่างกัน และความน่าเชื่อถือต่างกัน

RAID Level 0

- split data ในขนาดเท่าๆกัน เก็บไว้ในแต่ล่ะ disk
- ไม่มีการใช้ parity bits
- ไม่มีการใช้ redundancy

RAID Level 1

- มี copy ของข้อมูล
- วิธีนี้ใช้ได้ดีเมื่อต้องแคร์เรื่อง ประสิทธิภาพในการอ่านและความน่าเชื่อถือ มากกว่าพื้นที่จัดเก็บ

RAID Level 2

- ใช้ bit-level striping
- ใช้ Hamming code ในการตรวจสอบ error
- มี parity disk ต่อ disk ex. จากรูป Disk4-7 เป็น parity ของ Disk 0-3

RAID Level 3

- ใช้ byte-level striping
- ใช้ disk เดียวเป็น parity
- เป็นวิธีที่หายาก ไม่ค่อยเจอใครใช้จริง
- ลักษณะสำคัญคือไม่สามารถรองรับ request หลายอันพร้อมกันได้

RAID Level 4

- ใช้ block-level striping
- ใช้ disk เดียวเป็น parity
-

RAID Level 5

- ใช้ block-level striping

By Natthapach Anuwattananon

- ไม่เจาะจง disk ที่ใช้เป็น parity
- แบบนี้เป็นที่นิยมมาก เพราะราคาถูก

**Optical Disks**

CD-ROM Compact disk-read only memory

- อ่านได้อย่างเดียว

CD-R Compact

- เขียนได้จนกว่าจะเต็ม แต่ลบไม่ได้

CD-RW

- เขียนได้ ลบได้

DVD Digital Video Disk

- DVD-5 : 4.7 GB, DVD-9 : 8.9 GB, DVD-10 : 9.4 GB, DVD-18 : 17 GB

**Magnetic Tapes**

- very slow access time เพราะใช้ sequential access

**Storage Access**

- file ถูกแบ่งเป็นส่วนๆ ด้วยขนาดที่แน่นอน เรียกว่า block
- เวลาเก็บ(storage) ก็จะเก็บเป็น block เวลาส่งข้อมูล(data transfer) ก็จะส่งเป็น block เช่นกัน
- Database system ลดระยะเวลาในการส่งข้อมูลจาก disk ไปยัง Main memory ด้วยการเก็บ block จำนวนมากไว้ที่ main memory
- Buffer อยู่ใน main memory จัดเก็บ copy ของ block ที่อยู่ใน disk
- Buffer manager ระบบย่อย(subsystem) รับผิดชอบในการจัดสรรพื้นที่ของ buffer ใน main memory

**File Organization**

| | |
|---|---|
| database | คือการเก็บรวบรวมไฟล์ (collection of files) |
| file | คือลำดับของ *record* ที่เรียงต่อกัน |
| record | คือลำดับของ *field* ที่เรียงต่อกัน (แถว) |
| field | (คอลัมน์) |

**Fixed-Length Records**

- ตำแหน่งที่เก็บ record ที่ i คือ n * (i-1) เมื่อ n = ขนาดของแต่ล่ะ record
- การลบ record ทำได้หลายแบบ, เมื่อจะลบ record ที่ i และ n คือจำนวน record ทั้งหมด
    - เลื่อน i+1, ..., n ไปยัง i, ..., n-1 (ลบแล้วเลื่อนลงมา
    - ย้าย n ไปแทนที่ i (ลบแล้วเอาสุดท้ายมาแทน)
    - ไม่ต้องเลื่อน แต่ใช้ free lists

**Free Lists**

- มี record “header” เก็บ address ของ record แรกที่ถูกลบไปแล้วว่างอยู่
- record แรกจะเก็บที่อยู่ของ record ที่สอง และเก็บไปเรื่อยๆ จนเก็บ null เมื่อหมด

**Variable-Length Records**

- แต่ล่ะ record มี field ไม่เท่ากัน เกิดขึ้นได้จากหลายสาเหตุ เช่น
    - file นึง เก็บ record หลายประเภท
    - record ยอมให้บาง field มีหลายขนาดได้
    - record ยอมให้มีการซ้ำกันของ field

Slotted Page Structure โครงสร้างที่ใช้จัดการ variable-length records

* ดูรูปในชีทหน้า 11.33

- ประกอบด้วย header, free space, records
- Slotted page header มีไว้สำหรับเก็บ
    - จำนวนทางเข้า(entry) ของข้อมูลที่เก็บ record
    - จุดสิ้นสุดของ free space
    - ตำแหน่งและขนาดของแต่ล่ะ records
- ระหว่าง record จะไม่มีช่องว่าง

**Organization of Record** การจัดการ record มี 4 วิธี

Heap

- record เก็บไว้ตรงไหนใน file ก็ได้ที่ว่าง

Sequential

- เก็บ record ไว้เป็นลำดับ โดยลำดับขึ้นอยู่กับค่าที่จะใช้ในการค้นหา (search key) ของแต่ละ record

Hashing

- ใช้ hash function คำนวณค่าบางอย่างของ record
- ผลลัพธ์จาก function คือ block ของ file ที่เก็บ record นั้น

Multitable clustering file organization

- ปกติแล้ว record ที่มีความสัมพันธ์ต่างกัน ควรจะเก็บไว้นละ file
- แต่ด้วยการเก็บแบบ Multi.. นี้จะสามารถเก็บ record ที่มีความสัมพันธ์ต่างกัน ไว้ด้วยกันได้

* เดาว่า ความสัมพันธ์(relation) คือ field เหมือนกัน

**Sequential File Organization**

- เรียง record ด้วย search-key

By Natthapach Anuwattananon

- deletion ใช้ pointer chains (น่าจะเปลี่ยน pointer แบบ linked-list)
- insertion
    - ถ้ามี free space จะเพิ่มข้อมูลใหม่เข้าไปเลย
    - ถ้าไม่มี จะไปเก็บไว้ที่ *overflow block* แทน
    - ทั้งสองกรณีจะต้องทำการโยง pointer ใหม่
- ต้องการการจัดระเบียบใหม่ (reorganize) เป็นครั้งคราว เพื่อฟื้นฟูการเรียงลำดับ (sequential order)

**Record Representation**

- fixed-length fields จะมีวิธีเก็บคล้ายๆ ในการเขียนโปรแกรม
- variable-length fields จะเก็บด้วย offset, length

    offset = ตำแหน่งของ field, length = ขนาดของ field

- All fields start at predefined location, but extra indirection required for variable length fields

**Fixed-Length Representation**

fixed-length record

Reversed space

- สามารถใช้ได้เมื่อรู้ maximum length
- field ที่ไม่ได้ใช้จะแทนด้วย null หรือ end-of-record symbol (ตัว T กลับหัว)

Pointer Method

* ดูรูปหน้า 11-48,49

- สามารถใช้ได้แม้ไม่รู้ maximum length
- เก็บด้วย fixed-length แล้วเชื่อมกันด้วย pointer
- แต่มีข้อเสียคือ จะเสียพื้นที่ของทุก record ไปโดยที่ไม่ทำอะไรเลย ยกเว้น record แรกของ chain
- วิธีแก้คือแบ่ง block ใน file ออกเป็น 2 ประเภท
    - Anchor block เก็บ record แรกของ chain
    - Overflow block เก็บ record ที่เหลือ

# File Structure II

**Physical Records** ขึ้นอยู่กับขนาดของ sector (correspond to the size of a sector)

**Logical Records** ขึ้นอยู่กับการแบ่งข้อมูลแบบเรื่องที่แล้วมั้ง (correspond to natural divisions within the data)

## Operating System and File Access

- OS จะดึงข้อมูลจาก mass storage (secondary storage) ด้วยหน่วย(in unit of) Physical Record
- ข้อมูลที่ดึงมา จะเก็บไว้ใน buffer ซึ่งอยู่ใน Main memory
- Application จะที่จะใช้ข้อมูล จะดึงข้อมูลไปด้วยหน่วย Logical Record หรือ fields

## Physical Records and Buffer

- การจัดการ file ในเทอมของ block จะจัดการโดย OS
- OS จะรับผิดชอบในการอ่าน (reading request) ด้วยการอ่าน physical records แล้ววางไว้ใน buffer และ making the buffer available to the application
- การเก็บข้อมูล (storing) OS จะเก็บไว้ที่ buffer ก่อน จนกระทั่งสะสม physical record เสร็จสมบูรณ์ จึงจะส่งไปยัง mass storage

## File Descriptor

- จะเรียกว่า *file control block* ก็ได้
- เป็นตารางข้อที่เกี่ยวกับไฟล์ที่กำลังถูกจัดการ (the file being manipulated)
    - อุปกรณ์?
    - ชื่อไฟล์
    - ตำแหน่งบน buffer
    - มาร์คไว้ว่าจะเซฟไฟล์หลังจาก application จบลงหรือเปล่า

## Opening and Closing Files

- กระบวนการในการสร้างและทิ้ง file descriptor
- ในภาษาแบบ Imperative (Procedural)
    -
- ในภาษาแบบ OOP
    -

## Sequential Files

- เข้าถึงเป็นลำดับจากจุดเริ่มต้นไปยังจุดสุดท้าย
- ex. Audio, Video

* ต้องทดสอบ EOF (End of File)

By Natthapach Anuwattananon

```
while (not EOF)
        เรียก record ถัดไปใน file
```

**File Allocation Table (FAT)**

- เก็บเป็น *Cluster* (4-16 sector, ประมาณ 2 kB)
- FAT เก็บ record เป็น cluster
- ด้วยวิธี FAT, OS สามารถดึงข้อมูลจากไฟล์ cluster-by-cluster

**Detection of EOF** (การค้นหา EOF)

- EOF = End of file
- การตรวจสอบ EOF ทำได้ 2 วิธี
    - มี record พิเศษ เรียกว่า *sentinel*

```
while (record not sentinel)
        record = file.next()
```

    - ปล่อยให้เป็นหน้าที่ของ OS

```
while (not EOF)
        record = file.next()
```

**Key Field**

- field ที่ทำหน้าที่ในการระบุ record
- ex. เลขประจำตัว
- การจัดการไฟล์ด้วย key field สามารถลดเวลาการทำงานได้ดี

**Merging Two Sequential Files**

```
procedure mergeFile(inputA, inputB, output)          // สร้างฟังก์ชั่น
    if(A,B == EOF)
        return
```

**Text Files**

- sequential file ที่แต่ล่ะ record เก็บ encode character
- ex. ASCII file, Unicode file

**Editor vs Word Processors**

- Editor สร้างและแก้ไข text file
- Word Processor สามารถเพิ่มเติมโค้ดที่มองไม่เห็น (nonprintable code) เช่น การเปลี่ยน font, การจัดหน้ากระดาษ

By Natthapach Anuwattananon

# eXtensible Markup Language

## Programming Concerns

By Natthapach Anuwattananon