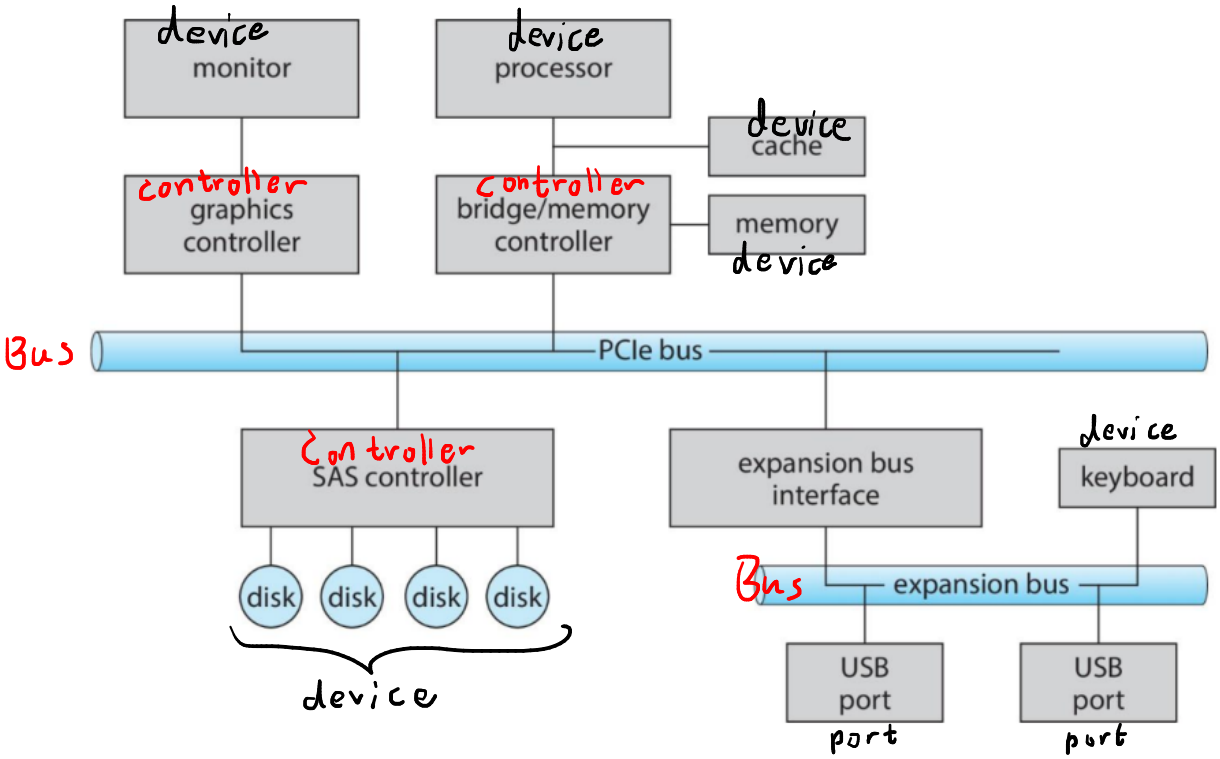
I/O Systems

ประกอบด้วย

- Storage (HDD, SSD, flash Drive)
- Transmission (LAN, Wi-fi, Bluetooth)
- Human - Interface (Printer, Mouse, keyboards)

Concept หลัก ของ I/O

- Port
- Bus → PCIe
- Controller → อุปกรณ์ควบคุม port, bus



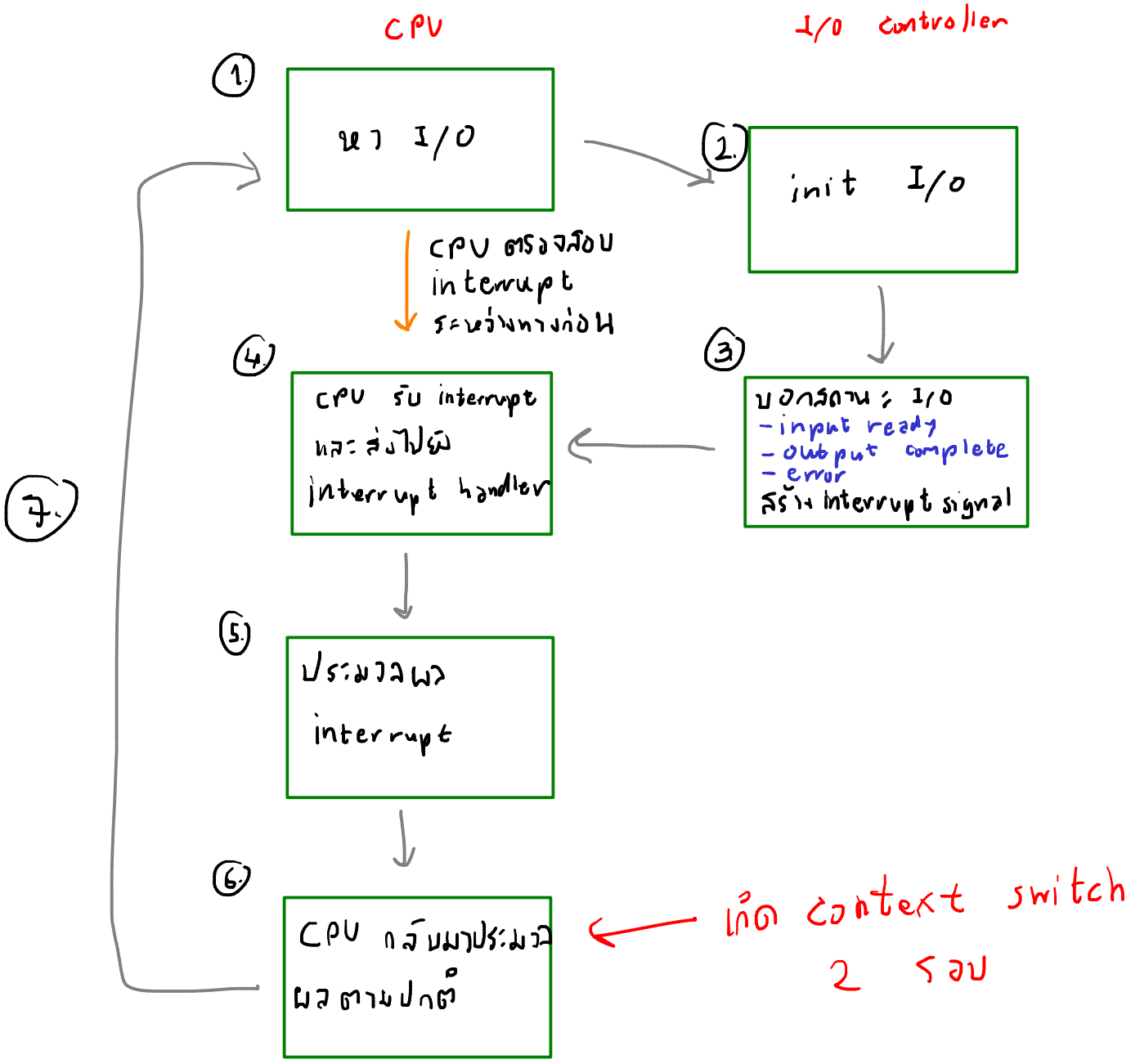
Polling

1.) อ่าน busy-bit จาก status register จนเป็น 0
2.) Host กำหนด ว่า read/write
3.) Host set → ready-bit
4.) controller set → busy-bit และ execute transfer
5.) controller clear → busy-bit, error-bit, command-ready bit เมื่อ transfer-done

หลังการเรียก I/O → Interrupts

- Polling - สามารถเกิดขึ้นได้ใน 3 instruction cycle
  → อ่าน status-bit
- CPU Interrupt-request line ← จาก I/O
- Interrupt Handler (ใช้รับ Interrupt)
- Interrupt Vector (ใช้ประมวลผลให้ handler ทำงานถูกต้อง)
  - context switch
  - priority

Interrupt-Driven I/O cycle



← เกิด context switch 2 รอบ

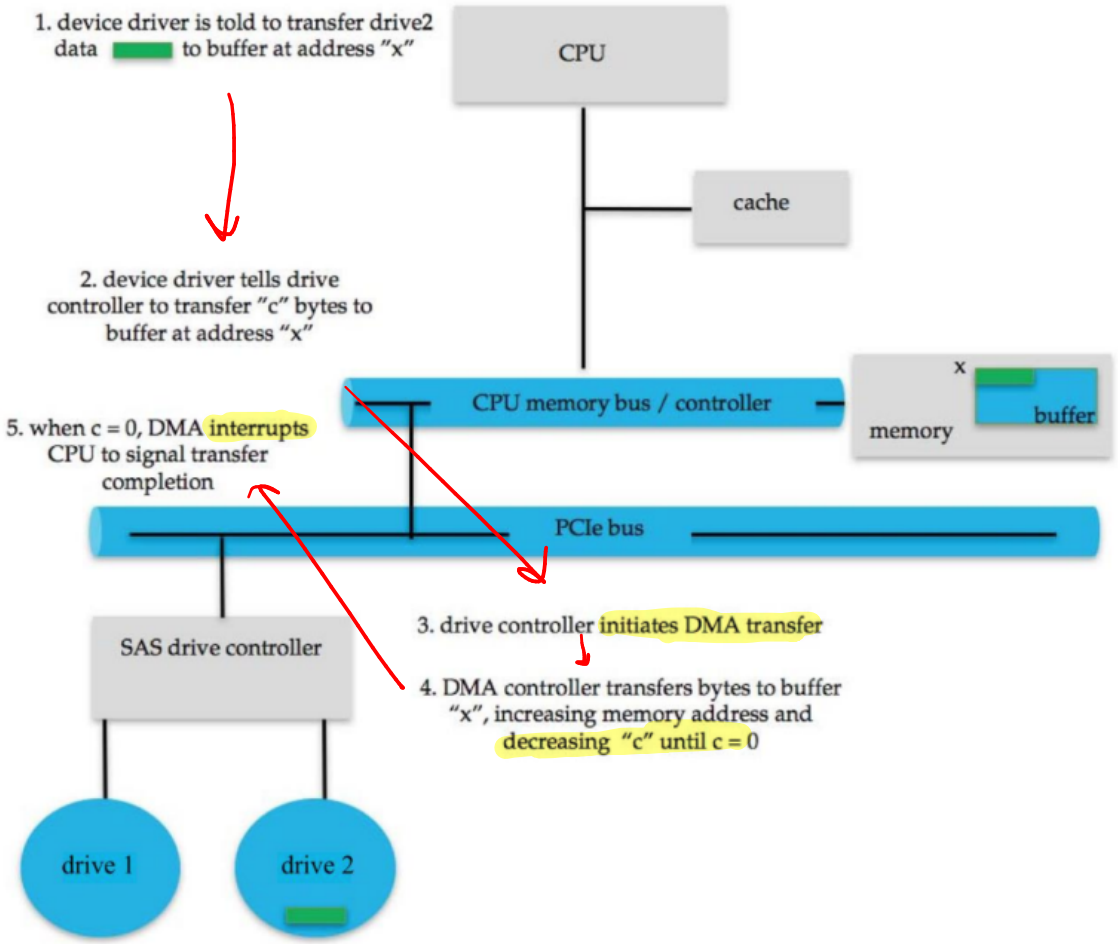
- การทำ Interrupt ยังใช้สำหรับ exception
  - Page Fault
  - trap
  - Multi-CPU สามารถรับ interrupt พร้อมกันได้ (ถ้า OS รองรับ)
  - ต้องเร็ว

โดยปกติ OS ใน desktop จะเกิด interrupt หลักร้อย / วินาที
Server จะเกิด interrupt หลักพัน / วินาที

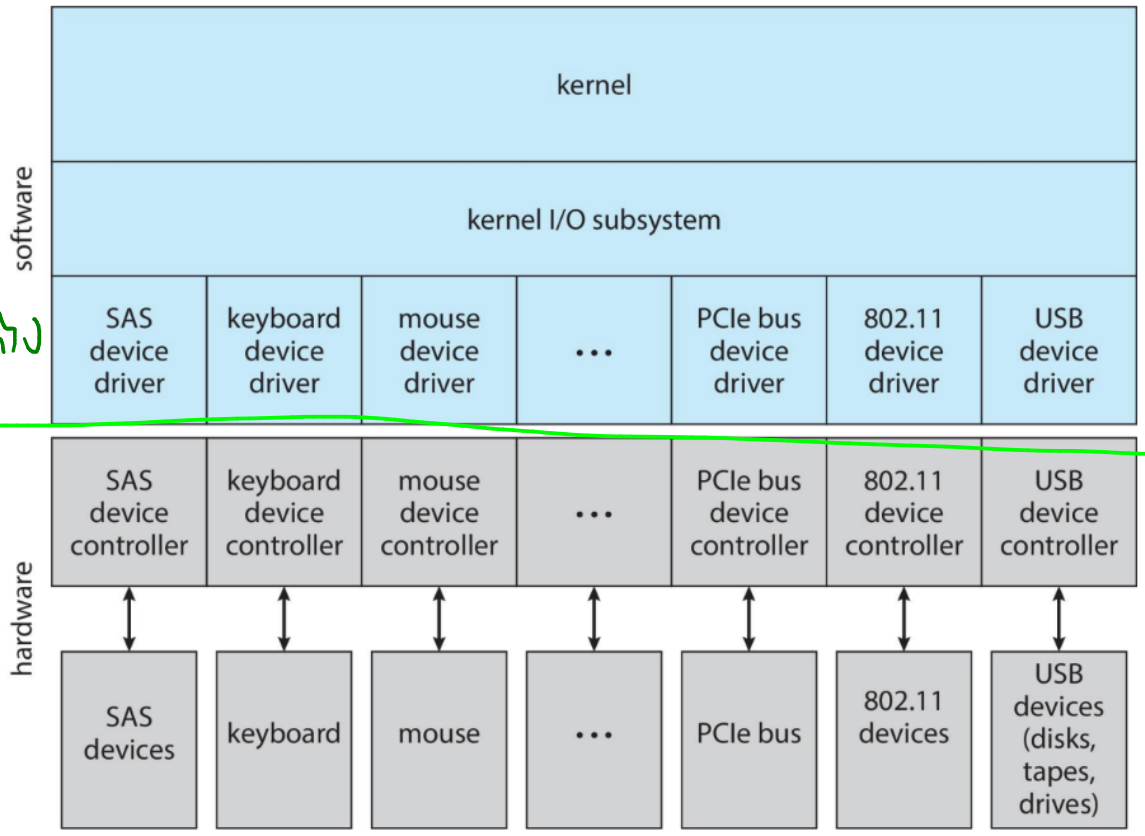
Direct Memory Access
- ต้องมี hardware controller
- ใช้สำหรับ I/O <-คุยกับ-> memory (RAM) โดยตรง
- โดย OS จะเขียน Command ต่างๆ ลงไปยัง memory
  - src → dest address
  - R / W mode
  - นับจำนวน bytes
  - ตำแหน่งที่จะเขียนลงไปยัง DMA controller
  - Bus Mastering (steal CPU cycle)
  - เมื่อคำสั่งใน DMA ทำงานเสร็จสิ้น → ส่ง interrupt

* ถ้าเป็น virtual address จะเรียกว่า DVMA

ขั้นตอนการทำงานของ DMA



Application I/O Interface



OS

- แตกต่างที่ของกัน

HW

ลักษณะของ I/O devices

aspect	variation	ex.
- data - transfer mode	char, block	terminal, disk
- access method	seq, rand	modem, CD
- transfer schedule	sync, async	tape, kbd
- sharing	dedicated, sharable	tape, kbd
- device speed	latency, seek time	HDD, SSD
- I/O direction	R, W, R/W	CD, GPU, disk

OS จะ Group I/O เป็น 4 ประเภท

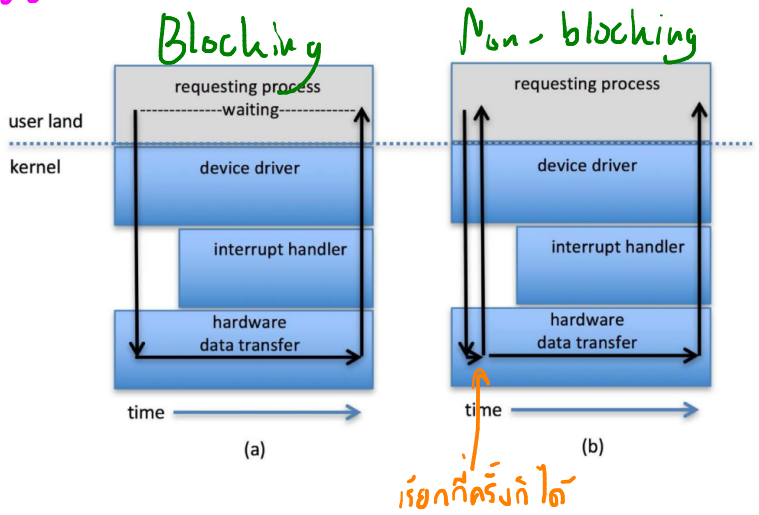
- Block I/O
  - disk drive
  - read, write, seek
  - DMA - อาจจะมี
- Character I/O
  - keyboard, mouse
  - getc(), putc()
- Memory-Mapped File access
- Network Socket
  - ใช้ socket interface
  - select()
  - สามารถสร้างได้เองแบบ < pipes, FIFOs, streams, queues, mailboxes

Clock & Timers

- มี current time
  elapsed time
  timer

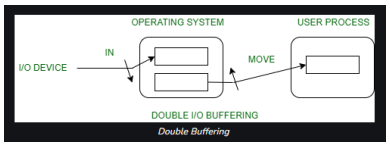
ioctl()

- Blocking I/O → process หยุดทำงานจนกว่า I/O เสร็จ
- Non-Blocking I/O → I/O เรียกเท่าไหร่ก็ได้
  - ใช้ใน multi-thread
- Asynchronous → process ทำงานระหว่าง I/O ทำงานไปด้วย
  - ใช้ยาก

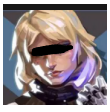


Vectored I/O
- ช่วยทำให้ 1 system call เรียกหลายๆ I/O ขึ้นมาได้
- ช่วยลด context switching และจำนวน System call ที่มากเกินไป

Kernel I/O Subsystem
- Scheduling
- Buffering - ใช้เก็บ data ใน memory ในขณะที่ส่งข้อมูลระหว่างอุปกรณ์
  - cope speed / transfer size ที่แตกต่างกัน
  - Double Buffering



- Caching - ใช้ในการ copy [เร็ว]
- Spooling - ใช้ hold output ex. printer
- Device Reservation - ให้ vip ในการเข้าถึง device
  ↓
  ทำให้เกิด deadlock ได้



Error Handling

- OS สามารถ Recover จากความผิดพลาดของ I/O ได้

  เช่น - Retry

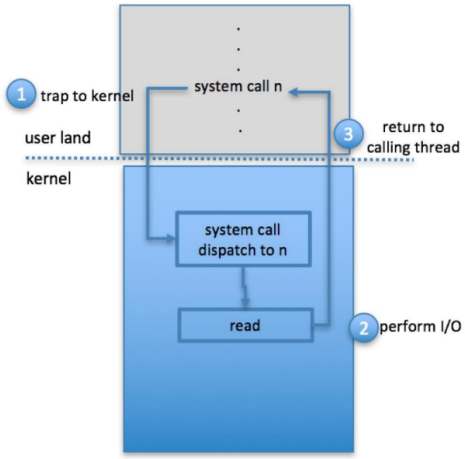
  - Return Error Code

  - บันทึกลง Log

I/O protection

- I/O ต้องเรียกผ่าน system calls เท่านั้น

  (user ไม่สามารถทำได้)



Kernel Data Structures

- ใช้เก็บ state ของ I/O
- มีหลายรูปแบบในการเก็บ
- ex. Windows ใช้ message passing

Power Management

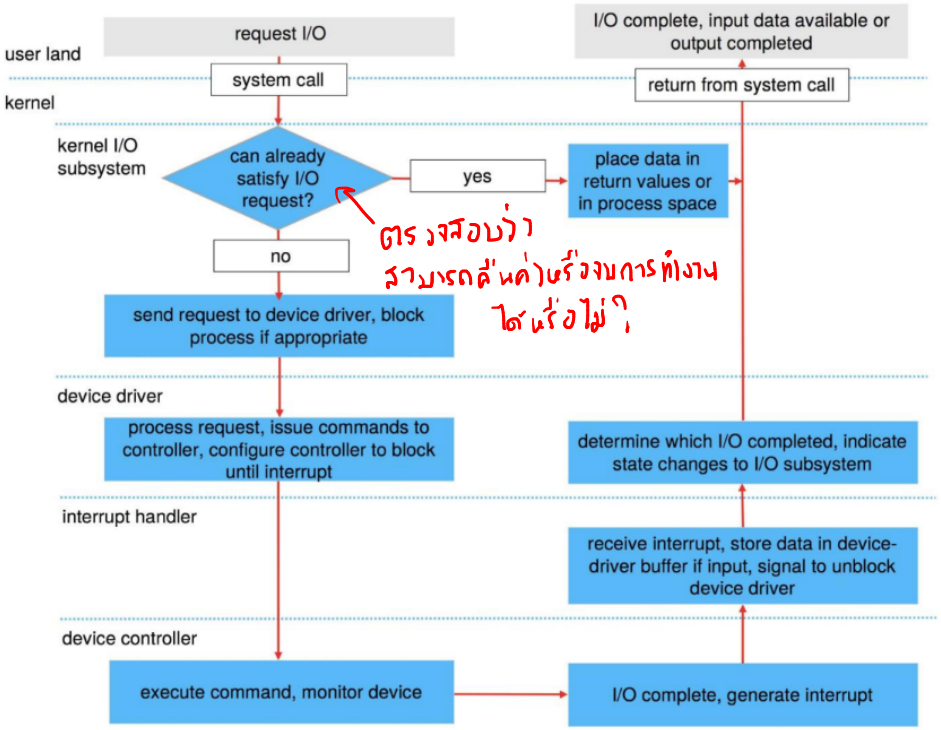
- ใช้ cooling ในการลดความร้อน
- ใน mobile device เป็น OS First-class aspect

ex. Android

- สร้าง tree
- อันไหนไม่ได้ใช้ → ปิด
- ถ้าใน tree ไม่ได้ใช้หมดแล้ว → ปิดหมด

- ปัจจุบันมี ACPI (Advance Configuration and power interface)

Life Cycle of An I/O Request



ตรวจสอบว่า สามารถดำเนินการต่อได้เร็วไม่?

Performance

- I/O เป็นส่วนหลักของความเร็วในระบบ
  - CPU execute driver, kernel I/O code
  - จำนวน context switching
  - Data Copying
  - โดยเฉพาะ: Network traffic

Performance Improving

- ลดจำนวน context switches
- ลดการ copy data
- ลด interrupts ด้วยการ transfer ไฟล์ขนาดใหญ่, ควบคุม controller
- ใช้ DMA
- ใช้ HW ใหม่ๆ
- Balance Spec ให้ได้ throughput ที่เหมาะที่สุด
- เอา user-mode process ไปยัง kernel threads /daemons

File - System

File F ประกอบด้วย

- Name
- Identifier
- Type
- Location
- Size
- Protection
- Datetime และ user identification
  - ข้อมูลในไฟล์

ไปเก็บส่วนหนึ่งของ Directory Structure → เป็น node เก็บข้อมูลในทุกไฟล์

File Operation

- create
- Write
- Read
- Reposition
- Delete
- Truncate
- Open (F_i) เอา content โยกจาก disk ไป memory
- Close (F_i) โยกกลับเข้า disk

ในการเปิดไฟล์ จะมีการจัดการดังนี้ เพื่อเปิด File

- open file table
- file pointer
- file open count

File Locking

- Shared Lock → ใช้อ่าน
- Exclusive Lock → ใช้เขียน

ประเภทการ lock

- mandatory
  - บังคับ
- Advisory
  - แนะนำ

File Structure

- none ← word, bytes
- simple record structure ← csv, \n \r
- complex structure ← xml, json

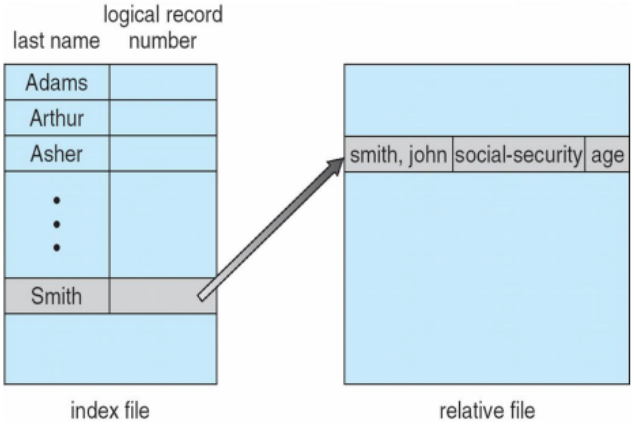
Access Method

- Sequential Access (คล้ายๆอ่านเทป)
  - read next
  - write next
  - reset ← เริ่มใหม่
- Direct Access
  - read n
  - write n
  - position to n ← ไปที่ไหนก็ได้
    - read next
    - write next
    - rewrite n

* n = relative block number.

วิธีอ่านไฟล์แบบอื่นๆ

- Index and Relative file



(Database?)

Disk Structure

- Disk สี่แบบ Partitions ได้
- RAID
- Disk อาจมี FS (File System) หรือ ไม่มีเลยก็ได้

ประเภทของ File System

Solaris

- tmpfs – memory-based volatile FS for fast, temporary I/O
- objfs – interface into kernel memory to get kernel symbols for debugging
- ctfs – contract file system for managing daemons
- lofs – loopback file system allows one FS to be accessed in place of another
- procfs – kernel interface to process structures
- ufs, zfs – general purpose file systems

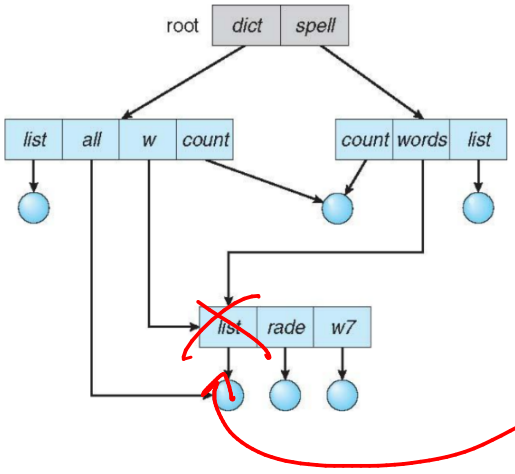
คำสั่งที่สามารถใช้ใน directory

- หา File (find)
- สร้าง (create)
- ลบ (delete)
- list
- rename
- traverse

การจัดการ Directory

- Single-level directory → ทุก user ใช้ directory เดียว
  → มีปัญหา Naming, Grouping
- Two-level directory → 1 user 1 directory
  → search เร็วขึ้น
  - มี path name
  แต่ ยังไม่มี grouping
- Tree-Structure directory
- Acyclic-Graph Directories – มี file และ subdirectory ที่ shared ร่วมกัน

Acyclic - Graph Directories (เพิ่มเติม)



- มี aliasing (dict/w/list
spell/words/list)

↓

ถ้าลบ list ออกไป ex. ลบ w/list
จะทำให้เกิด dangling pointer

แก้ได้โดยทำ Back pointer
แล้วสั่งลบ pointers ทีหมด

→ เกิด directory entry type ใหม่

→ Link ← เหมือน shortcut ไปยังไฟล์ที่มีอยู่

[Resolve the Link ใช้ในการ follow pointer
ไปยังไฟล์

General Graph Directory

→ จะรู้ได้ยังไงว่าไม่มี cycle ใน Graph แน่ๆ

Allow → Link File ได้อย่างเดียว

→ Garbage Collection

→ เมื่อมีการ link ให้ run cycle detection algo.

Current Directory ทำอะไรได้บ้าง

→ pwd

→ create / delete a file
touch / rm

Protection

→ R W X
Read Write Excute

→ Append, delete, list

binary 3 bit ←

Owner	Group	Other
r w x	r - x	r - x
4+2+1	4+0+1	4+0+1
7	5	5

* อย่า 777