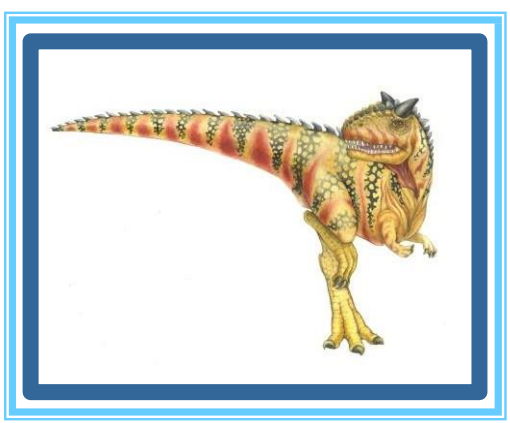
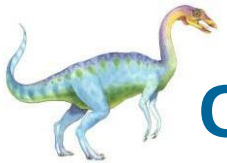
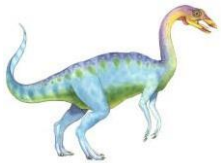
Chapter 8: Virtual-Memory Management





Chapter 8: Virtual-Memory Management

- Background
- Demand Paging เพจความต้องการ
- Page Replacement การเปลี่ยนหน้า
- Allocation of Frames การจัดสรรเฟรม



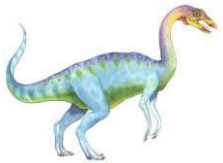
Objectives

เพื่ออธิบายประโยชน์ของระบบหน่วยความจำเสมือน

- To describe the benefits of a virtual memory system

เพื่ออธิบายแนวคิดของเพจความต้องการ อัลกอริธึมการแทนที่เพจ และการจัดสรรเฟรมเพจ

- To explain the concepts of demand paging, page-replacement algorithms, and allocation of page frames



Background

โค้ดจำเป็นต้องอยู่ในหน่วยความจำเพื่อดำเนินการ แต่แทบไม่ได้ใช้ทั้งโปรแกรม

- Code needs to be in memory to execute, but entire program rarely used
  - รหัสข้อผิดพลาด กิจวัตรที่ผิดปกติ โครงสร้างข้อมูลขนาดใหญ่
    - Error code, unusual routines, large data structures
- ไม่จำเป็นต้องใช้รหัสโปรแกรมทั้งหมดในเวลาเดียวกัน
  - Entire program code not needed at same time
- พิจารณาความสามารถในการรันโปรแกรมที่โหลดเพียงบางส่วน
  - Consider ability to execute partially-loaded program
    - โปรแกรมไม่ถูกจำกัดด้วยขีดจำกัดของหน่วยความจำกายภาพอีกต่อไป
      - Program no longer constrained by limits of physical memory
    - แต่ละโปรแกรมใช้หน่วยความจำน้อยลงในขณะทำงาน -> มีโปรแกรมทำงานมากขึ้นในเวลาเดียวกัน
      - Each program takes less memory while running -> more programs run at the same time
        - เพิ่มการใช้งาน CPU และปริมาณงานโดยไม่เพิ่มเวลาตอบสนองหรือเวลาตอบสนอง
          - Increased CPU utilization and throughput with no increase in response time or turnaround time
    - I/O น้อยลงที่จำเป็นในการโหลดหรือสลับโปรแกรมลงในหน่วยความจำ -> แต่ละโปรแกรมของผู้ใช้จะทำงานเร็วขึ้น
      - Less I/O needed to load or swap programs into memory -> each user program runs faster

Background

หน่วยความจำเสมือน – การแยกหน่วยความจำลอจิคัลของผู้ใช้ออกจากหน่วยความจำกายภาพ

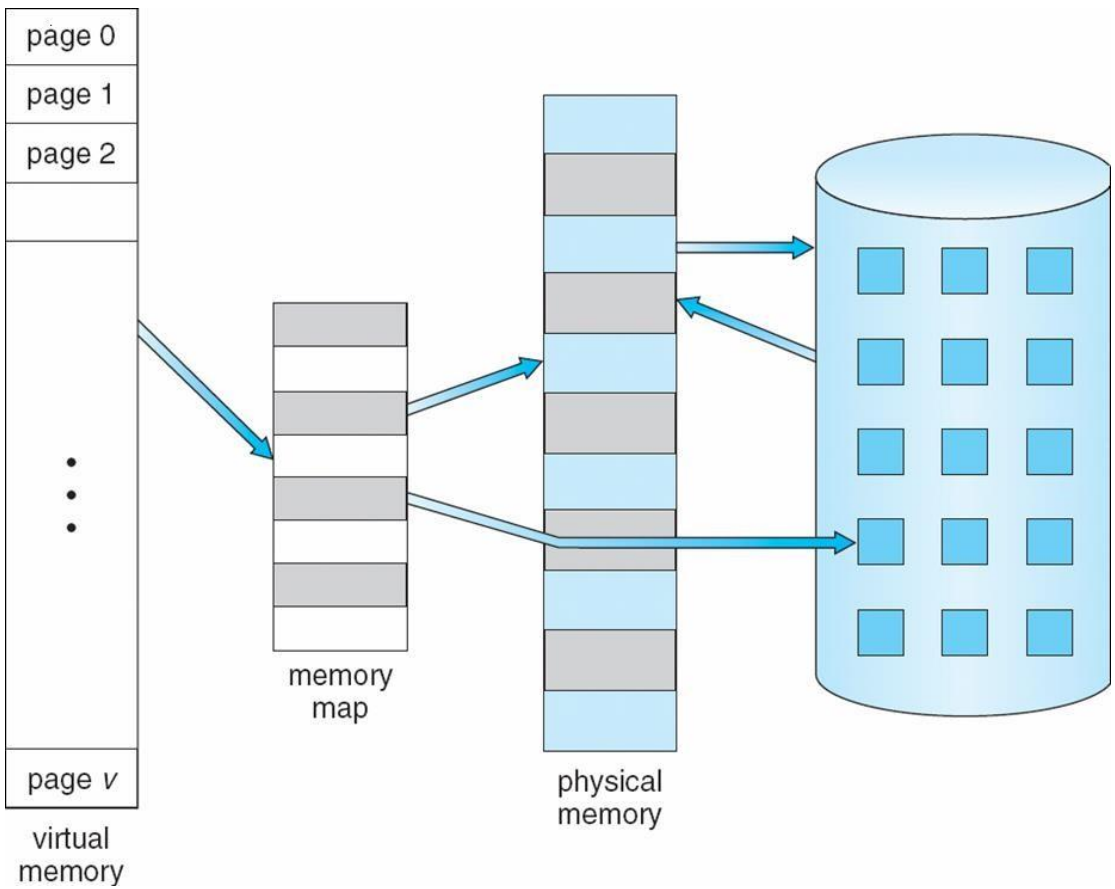
- **Virtual memory** – separation of user logical memory from physical memory.
  - เพียงส่วนหนึ่งของโปรแกรมเท่านั้นที่ต้องอยู่ในหน่วยความจำเพื่อดำเนินการ
    - Only part of the program needs to be in memory for execution
    - พื้นที่ที่อยู่แบบลอจิคัลจึงสามารถมีขนาดใหญ่กว่าพื้นที่ที่อยู่ทางกายภาพได้มาก
    - Logical address space can therefore be much larger than physical address space
    - อนุญาตให้ใช้ช่องว่างที่อยู่ร่วมกันโดยหลายกระบวนการ
    - Allows address spaces to be shared by several processes
    - ช่วยให้การสร้างกระบวนการมีประสิทธิภาพมากขึ้น
    - Allows for more efficient process creation
    - โปรแกรมเพิ่มเติมที่ทำงานพร้อมกัน
    - More programs running concurrently
    - ต้องใช้ I/O น้อยลงในการโหลดหรือสลับกระบวนการ
    - Less I/O needed to load or swap processes

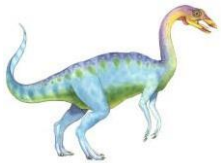
  พื้นที่ที่อยู่เสมือน – มุมมองเชิงตรรกะของวิธีการจัดเก็บกระบวนการในหน่วยความจำ
- **Virtual address space** – logical view of how process is stored in memory
  - มักจะเริ่มต้นที่ที่อยู่ 0 ที่อยู่ติดกันจนกระทั่งสิ้นสุดช่องว่าง
    - Usually start at address 0, contiguous addresses until end of space
    - ในขณะเดียวกัน หน่วยความจำกายภาพที่ถูกจัดอยู่ในกรอบหน้า
    - Meanwhile, physical memory organized in page frames
    - MMU ต้องแมปตรรกะกับกายภาพ
    - MMU must map logical to physical

  หน่วยความจำเสมือนสามารถใช้งานได้ผ่าน:
- Virtual memory can be implemented via:
  - เพจความต้องการ
    - Demand paging
    - การแบ่งส่วนความต้องการ
    - Demand segmentation

หน่วยความจำเสมือนที่มีขนาดใหญ่กว่าหน่วยความจำกายภาพ

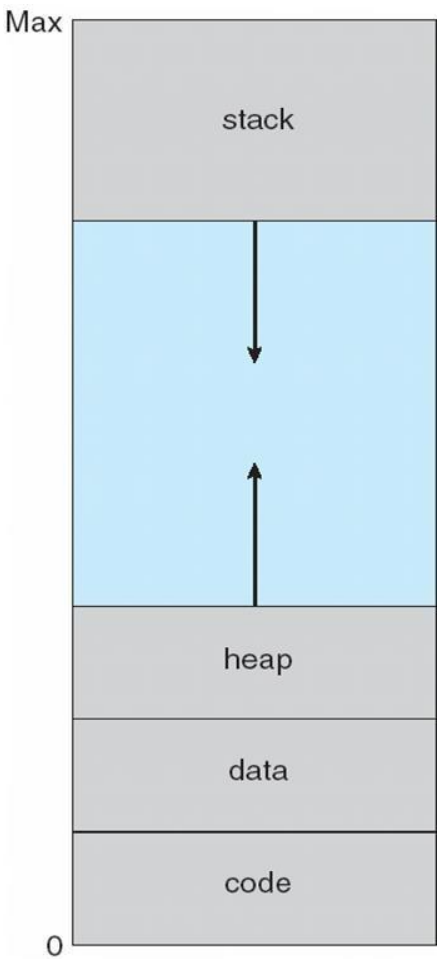
Virtual Memory That is Larger Than Physical Memory





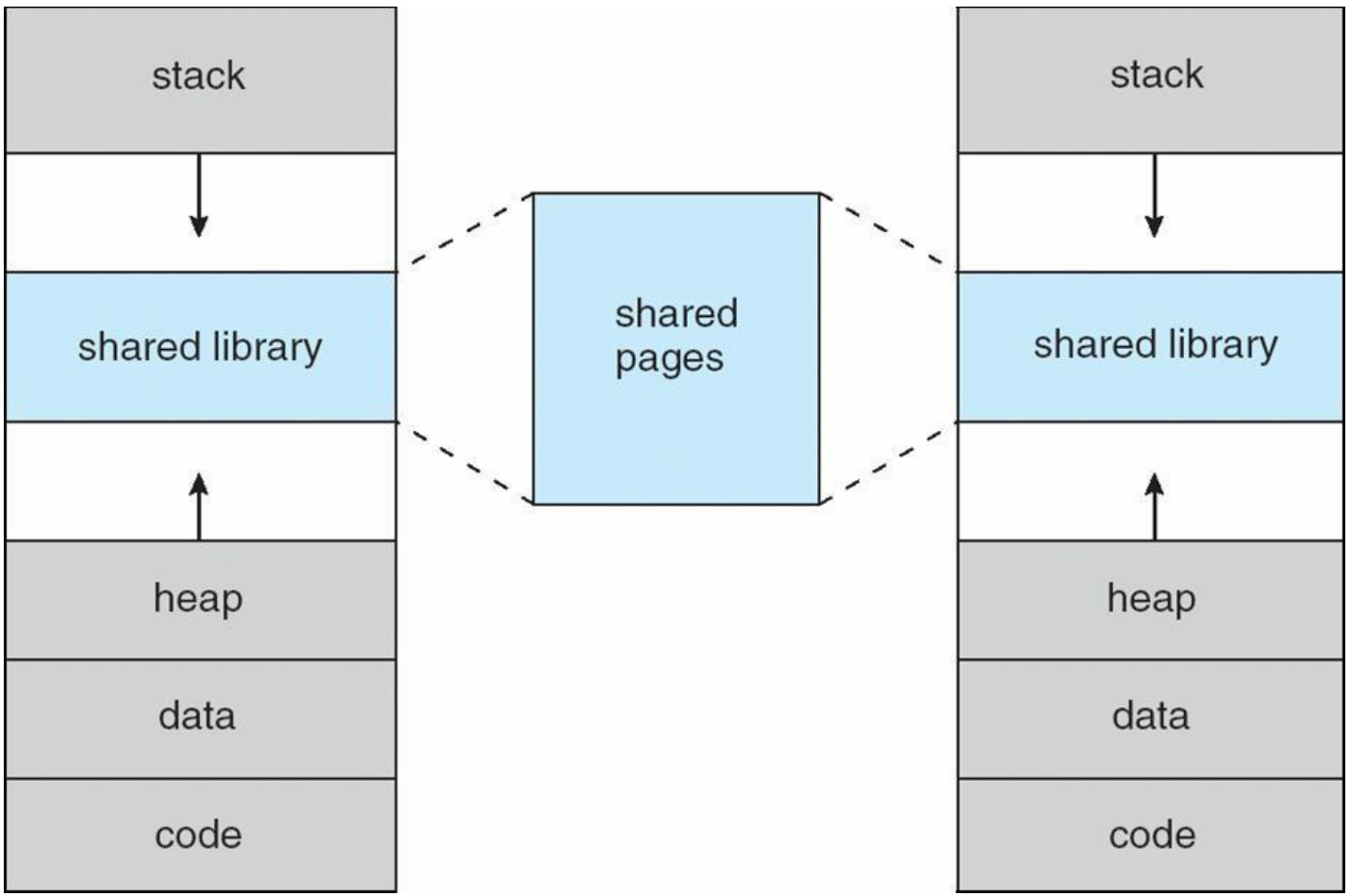
พื้นที่ที่อยู่เสมือน

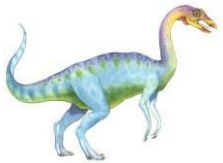
Virtual-address Space



ไลบรารีที่ใช้ร่วมกันโดยใช้หน่วยความจำเสมือน

Shared Library Using Virtual Memory





Demand Paging เพจความต้องการ

(การจัดสรร Paging ตามความต้องการที่ร้องขอ)

นำเพจเข้าสู่หน่วยความจำเมื่อจำเป็นเท่านั้น

- Bring a page into memory only when it is needed
  - ต้องการ I/O น้อยลง
  - Less I/O needed
  - ต้องการหน่วยความจำน้อยลง
  - Less memory needed
  - ตอบสนองเร็วขึ้น
  - Faster response
  - ผู้ใช้มากขึ้น
  - More users

จำเป็นต้องมีหน้า ⇒ อ้างอิงถึงหน้านั้น

- Page is needed ⇒ reference to it (หากต้องการใช้ page ให้อ้างตำแหน่งถึง page ที่ต้องการ)
  - การอ้างอิงไม่ถูกต้อง ⇒ ยกเลิก
  - invalid reference ⇒ abort (หากอ้างตำแหน่งไม่ถูกต้องให้ยกเลิก)
  - ไม่อยู่ในหน่วยความจำ ⇒ นำมาสู่หน่วยความจำ
  - not-in-memory ⇒ bring to memory (หากอ้างแล้วไม่มีใน memory ให้นำเข้ามาไว้ใน memory)

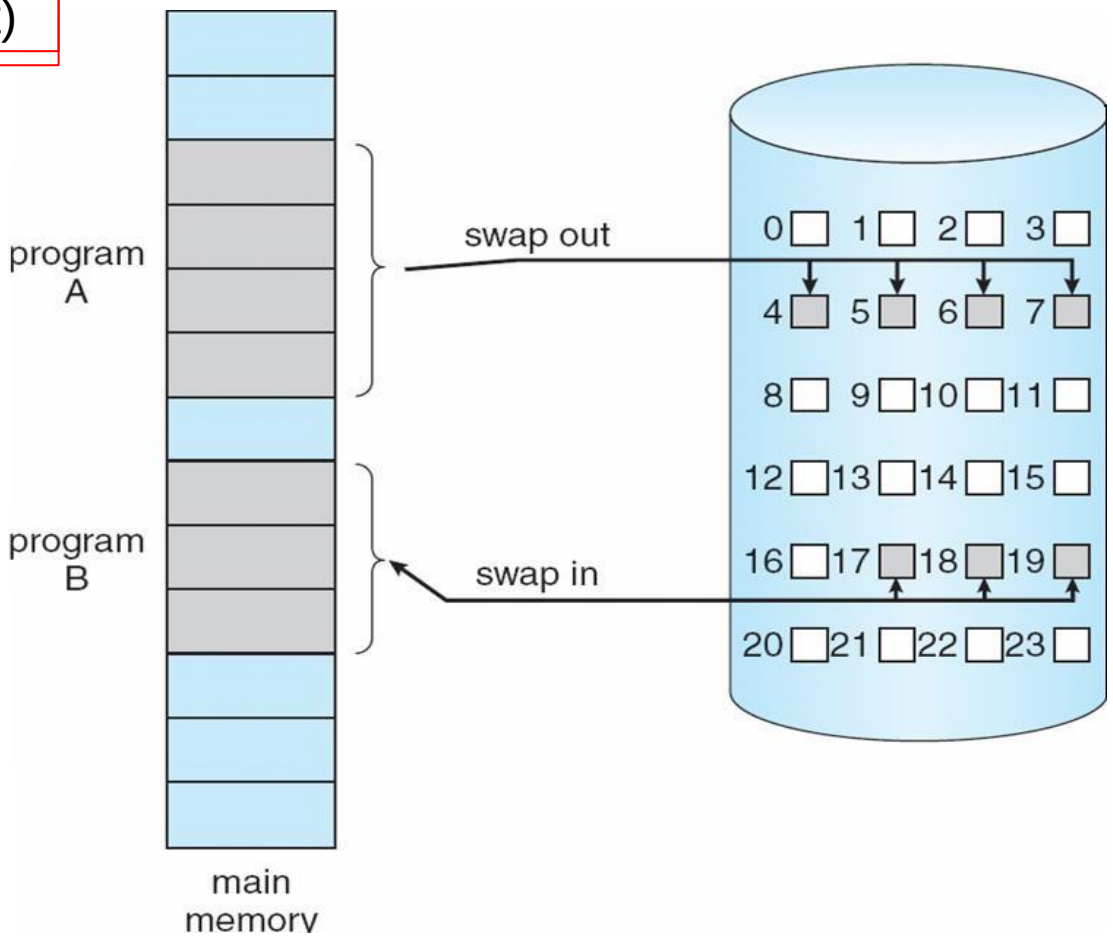
Lazy swapper – อย่าสลับหน้าลงในหน่วยความจำ เว้นแต่จำเป็นต้องใช้หน้า

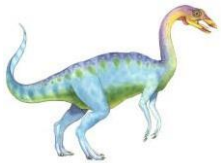
- **Lazy swapper** – never swaps a page into memory unless page will be needed
  - Swapper ที่เกี่ยวข้องกับเพจคือเพจเจอร์
  - Swapper that deals with pages is a **pager**

การถ่ายโอนหน่วยความจำแบบเพจไปยังพื้นที่ดิสก์ที่อยู่ติดกัน

Transfer of a Paged Memory to Contiguous Disk Space

มี pager เป็นตัวย้าย page
เข้า (swap in) หรือออก (swap out)





บิตที่ถูกต้อง-ไม่ถูกต้อง

Valid-Invalid Bit

แต่ละรายการในตารางเพจจะเชื่อมโยงบิตที่ถูกต้องและไม่ถูกต้อง (v ⇒ ในหน่วยความจำ, i ⇒ ไม่ได้อยู่ในหน่วยความจำ)

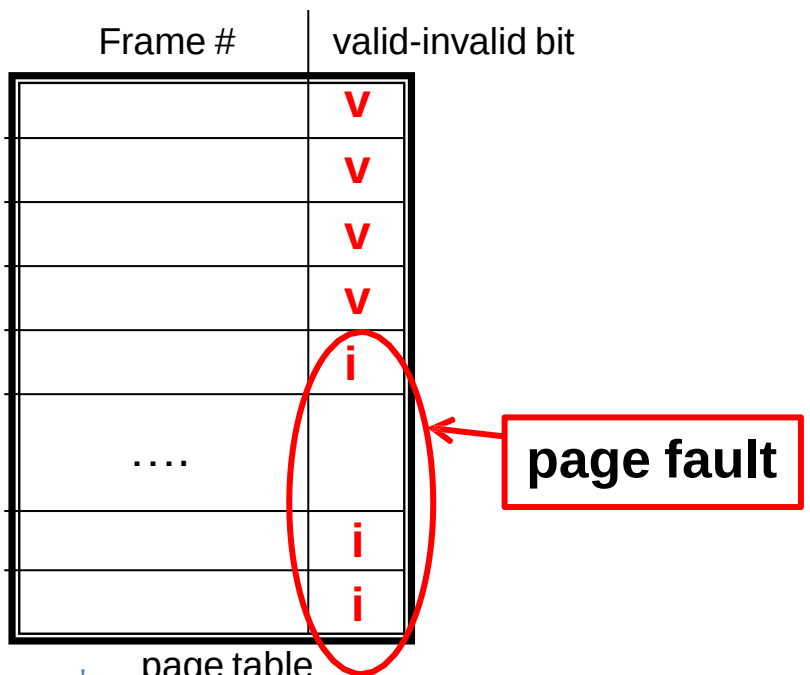
- With each page table entry a valid–invalid bit is associated (**v** ⇒ in-memory, **i** ⇒ not-in-memory)

บิตที่ถูกต้องในตอนแรกถูกตั้งค่าเป็น i ในทุกรายการ

- Initially valid–invalid bit is set to **i** on all entries

ตัวอย่างสแนปชอตตารางหน้า:

- Example of a page table snapshot:

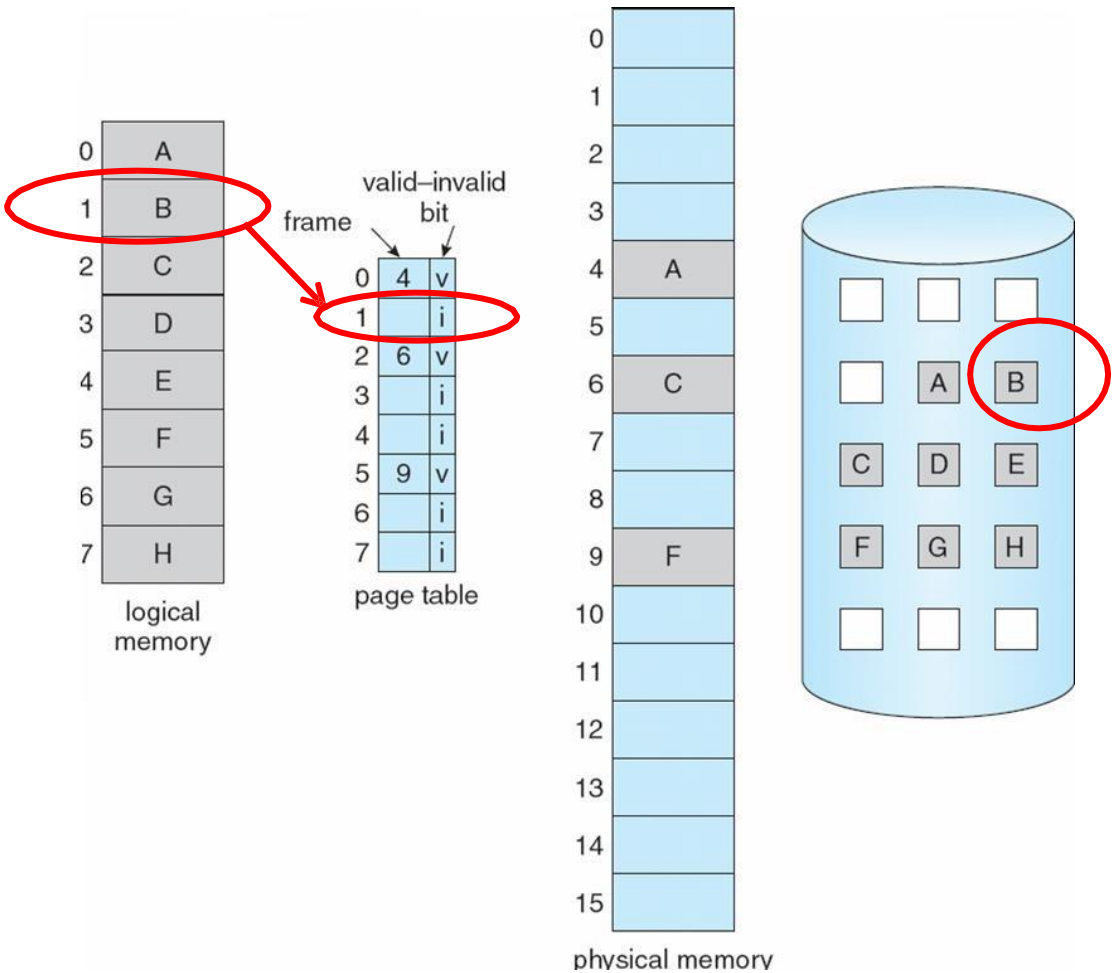


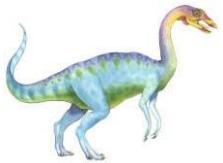
ในระหว่างการแปลที่อยู่ หากบิตที่ถูกต้อง–ไม่ถูกต้องในรายการตารางเพจคือ I ⇒ ข้อบกพร่องของเพจ

- During address translation, if valid–invalid bit in page table entry is **I** ⇒ page fault (การอ้างอิงผิดหน้าหรือไม่พบหน้าที่ต้องการ).

ตารางเพจเมื่อบางเพจไม่อยู่ในหน่วยความจำหลัก

Page Table When Some Pages Are Not in Main Memory





Page Fault ความผิดของหน้า

เมื่อมีการอ้างอิงผิดหน้าหรือไม่พบหน้าที่ต้องการในหน่วยความจำหลัก (page fault) จะมีขั้นตอนดาเนินการดังนี้

หากมีการอ้างอิงไปยังเพจ การอ้างอิงไปยังเพจนั้นเป็นครั้งแรกจะดักกับระบบปฏิบัติการ:

- If there is a reference to a page, first reference to that page will trap to operating system:

ความผิดของหน้า

page fault

ระบบปฏิบัติการจะพิจารณาตารางอื่นเพื่อตัดสินใจ:

1. Operating system looks at another table to decide:
   - การอ้างอิงไม่ถูกต้อง ⇒ ยกเลิก
     - Invalid reference ⇒ abort
   - แค่ไม่ได้อยู่ในความทรงจำ
     - Just not in memory

รับกรอบเปล่า

2. Get empty frame

สลับหน้าเป็นเฟรม

3. Swap page into frame

รีเซ็ตตาราง

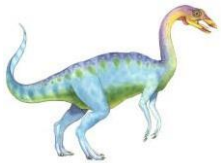
4. Reset tables

ตั้งค่าบิตการตรวจสอบ = v

5. Set validation bit = **v**

รีสตาร์ทคำสั่งที่ทำให้เกิดข้อผิดพลาดของหน้า

6. Restart the instruction that caused the page fault

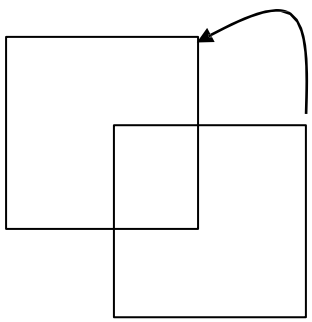


ความผิดพลาดของหน้า

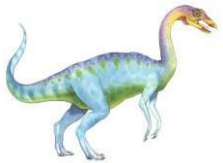
Page Fault (Cont.)

รีสตาร์ทคำสั่ง

- Restart instruction (ทำต่อจากจุดที่ได้ทำมาแล้วล่าสุด หรือ ทำต่อจากจุดที่เกิด page fault ขึ้น)
  - บล็อกย้าย
  - block move

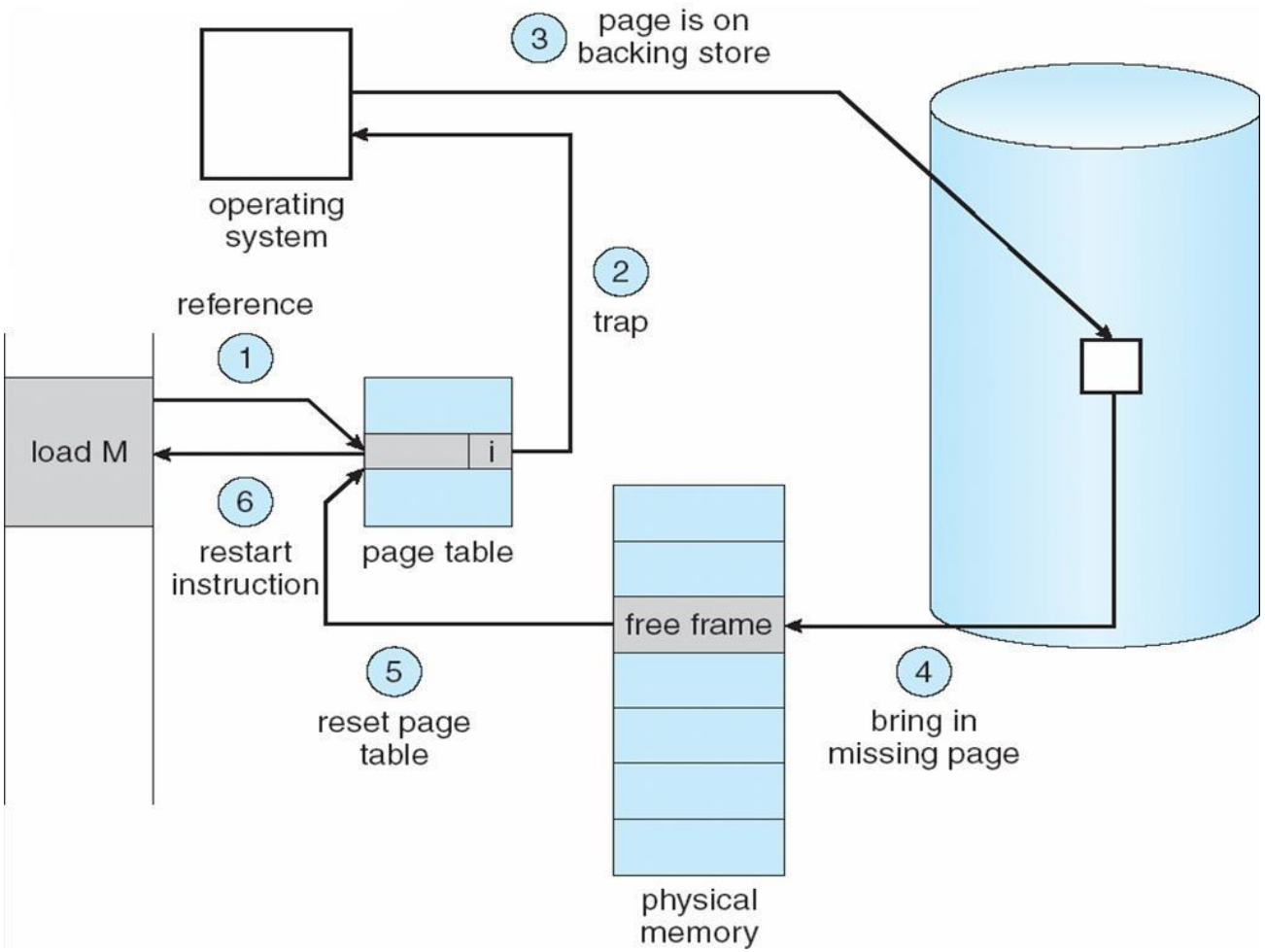


  - ตำแหน่งเพิ่ม/ลดอัตโนมัติ
  - auto increment/decrement location



ขั้นตอนในการจัดการกับ Page Fault

Steps in Handling a Page Fault



จะเกิดอะไรขึ้นหากไม่มีเฟรมฟรี?

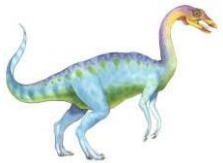
What happens if there is no free frame?

การแทนที่หน้า – ค้นหาหน้าบางหน้าในหน่วยความจำ แต่ไม่ได้ใช้งานจริง ให้สลับออก

- **Page replacement** – find some page in memory, but not really in use, swap it out
  - อัลกอริทึม
  - algorithm
  - ประสิทธิภาพ – ต้องการอัลกอริธึมที่จะส่งผลให้มีข้อผิดพลาดของหน้าน้อยที่สุด
  - performance – want an algorithm which will result in minimum number of page faults

หน้าเดียวกันอาจถูกดึงเข้ามาในความทรงจำหลายครั้ง

- Same page may be brought into memory several times



Page Replacement การเปลี่ยนหน้า

ป้องกันการจัดสรรหน่วยความจำมากเกินไปโดยการแก้ไขรูทีนการบริการข้อบกพร่องของเพจเพื่อรวมการแทนที่เพจด้วย

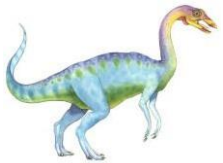
- **Prevent over-allocation of memory** by modifying page-fault service routine to include page replacement

ใช้บิตแก้ไข (สกปรก) เพื่อลดค่าใช้จ่ายในการถ่ายโอนเพจ - เฉพาะหน้าที่แก้ไขเท่านั้นที่จะถูกเขียนลงดิสก์

- Use **modify (dirty) bit** to reduce overhead of page transfers – only modified pages are written to disk

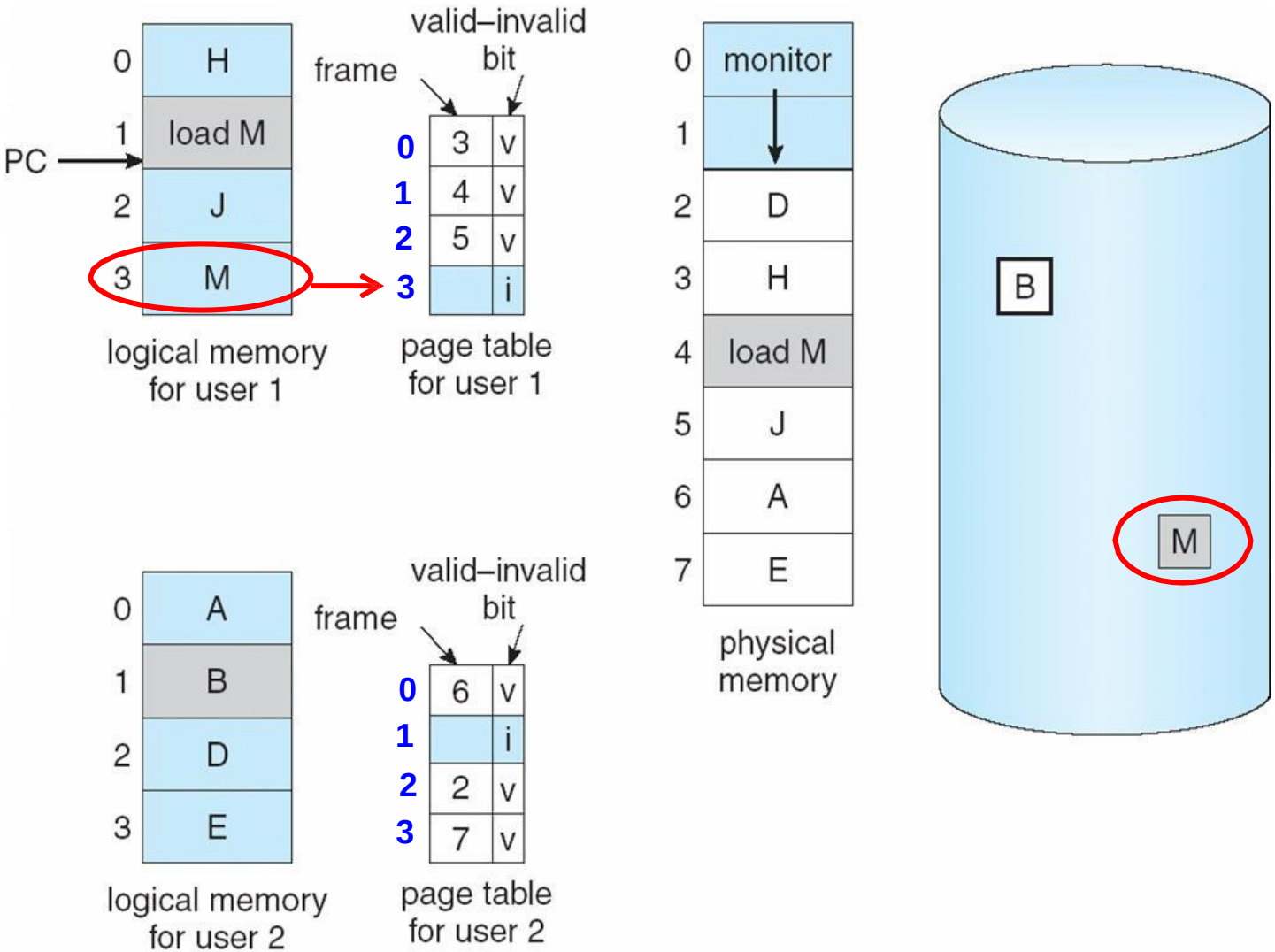
การแทนที่หน้าจะทำให้การแยกระหว่างหน่วยความจำลอจิคัลและหน่วยความจำกายภาพสมบูรณ์ - หน่วยความจำเสมือนขนาดใหญ่สามารถจัดเตรียมไว้ในหน่วยความจำกายภาพที่มีขนาดเล็กกว่าได้

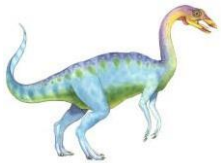
- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory



จำเป็นต้องเปลี่ยนหน้า

Need For Page Replacement





การเปลี่ยนหน้าพื้นฐาน

Basic Page Replacement

ค้นหาตำแหน่งของหน้าที่ต้องการบนดิสก์

1. Find the location of the desired page on disk

ค้นหาเฟรมฟรี:

2. Find a free frame: ถ้ามีกรอบว่างก็ใช้
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a **victim** frame

หากไม่มีเฟรมว่าง ให้ใช้อัลกอริธึมการแทนที่หน้าเพื่อเลือกเฟรมเหยื่อ

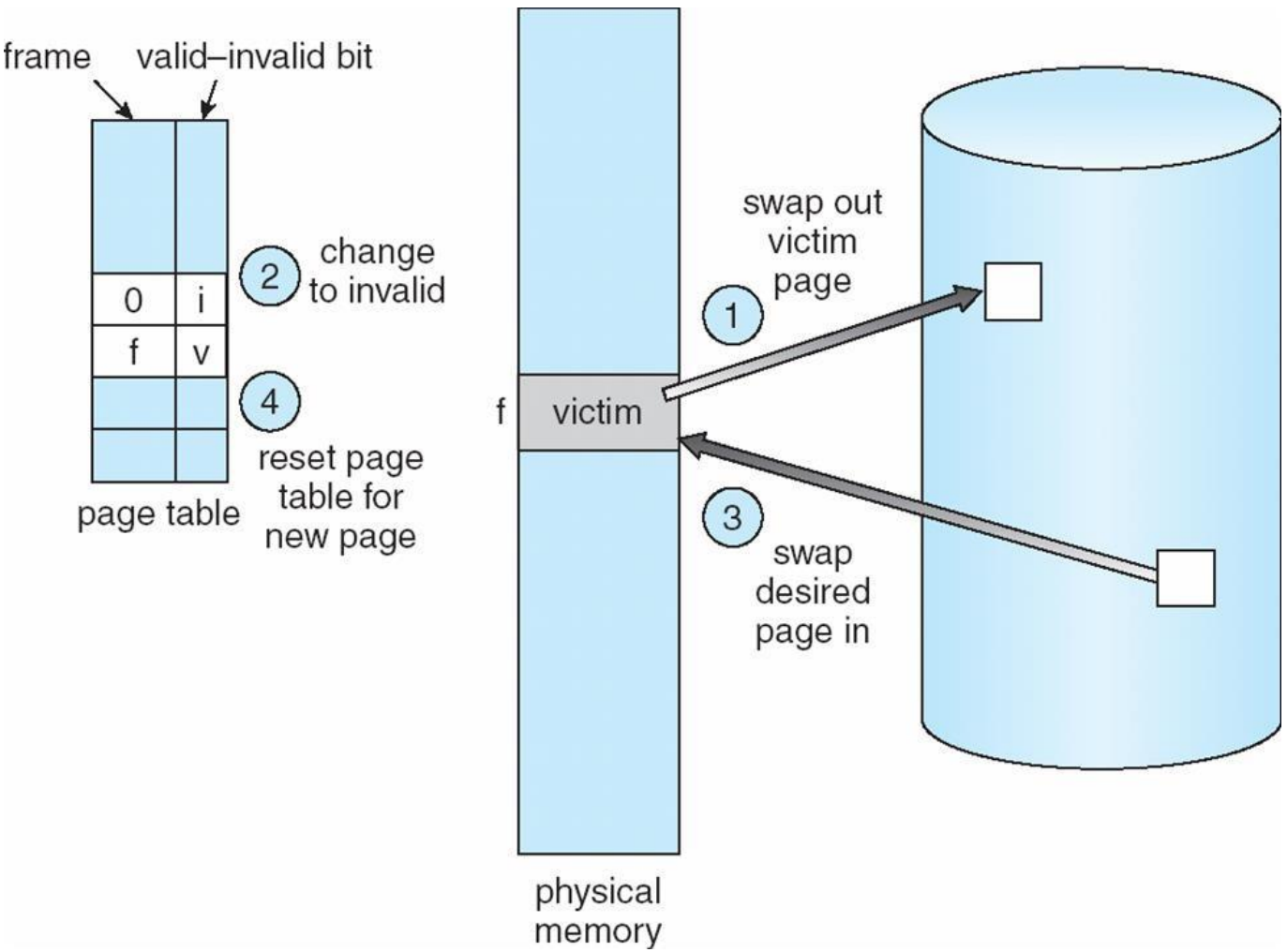
นำหน้าที่ต้องการมาไว้ในเฟรมอิสระ (ใหม่) อัพเดตตารางเพจและเฟรม

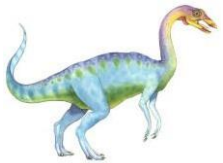
3. Bring the desired page into the (newly) free frame; update the page and frame tables

เริ่มกระบวนการใหม่

4. Restart the process

Page Replacement





อัลกอริธึมการแทนที่หน้า

Page Replacement Algorithms

ต้องการอัตราความผิดพลาดของหน้าต่ำสุด

- **Want lowest page-fault rate**

ประเมินอัลกอริทึมโดยการรันบนสตริงเฉพาะของการอ้างอิงหน่วยความจำ (สตริงอ้างอิง) และคำนวณจำนวนข้อบกพร่องของหน้าในสตริงนั้น

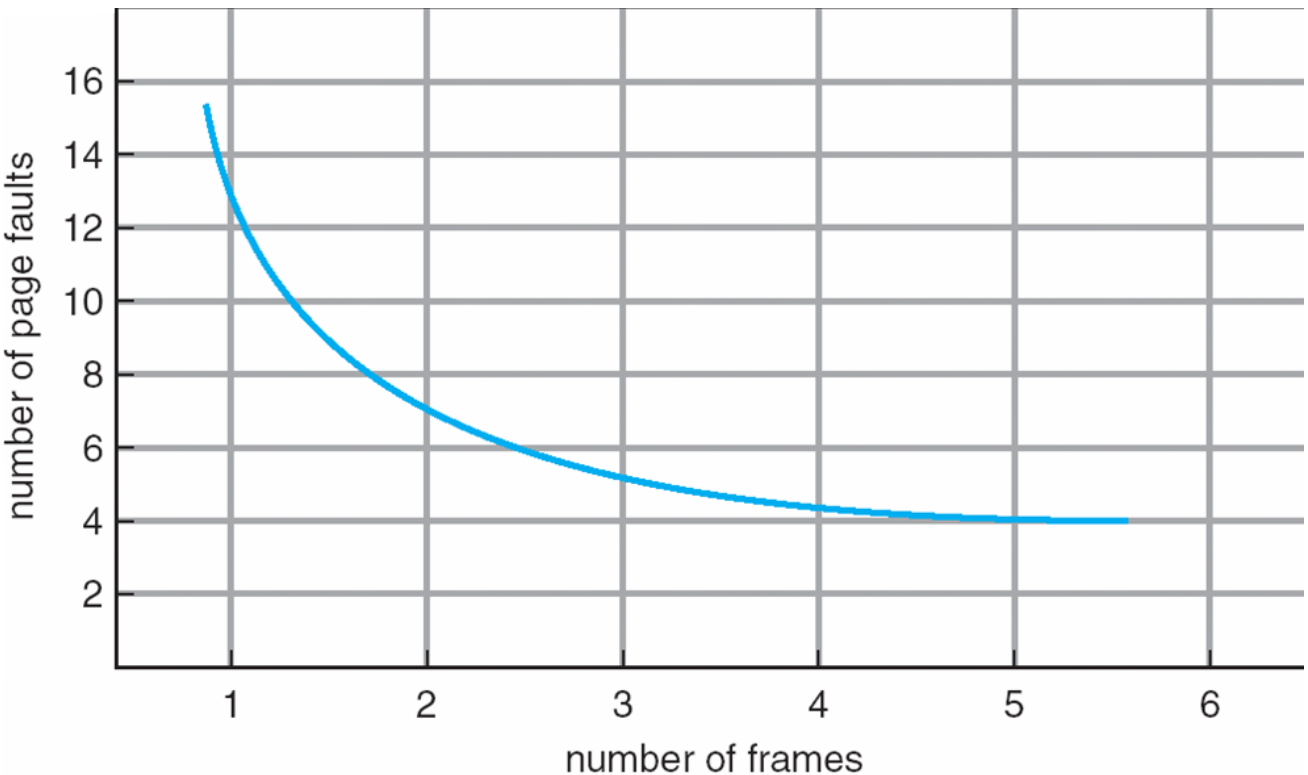
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string

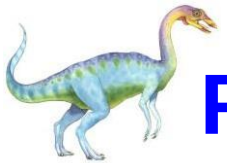
ในตัวอย่างทั้งหมดของเรา สตริงอ้างอิงคือ

- In all our examples, the reference string is

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

กราฟของ Page Faults กับจำนวนเฟรม
Graph of Page Faults Versus The Number of Frames





First-In-First-Out (FIFO) Algorithm

สตริงอ้างอิง: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

3 เฟรม (สามารถอยู่ในหน่วยความจำได้ 3 หน้าต่อครั้งต่อกระบวนการ)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

- 4 frames

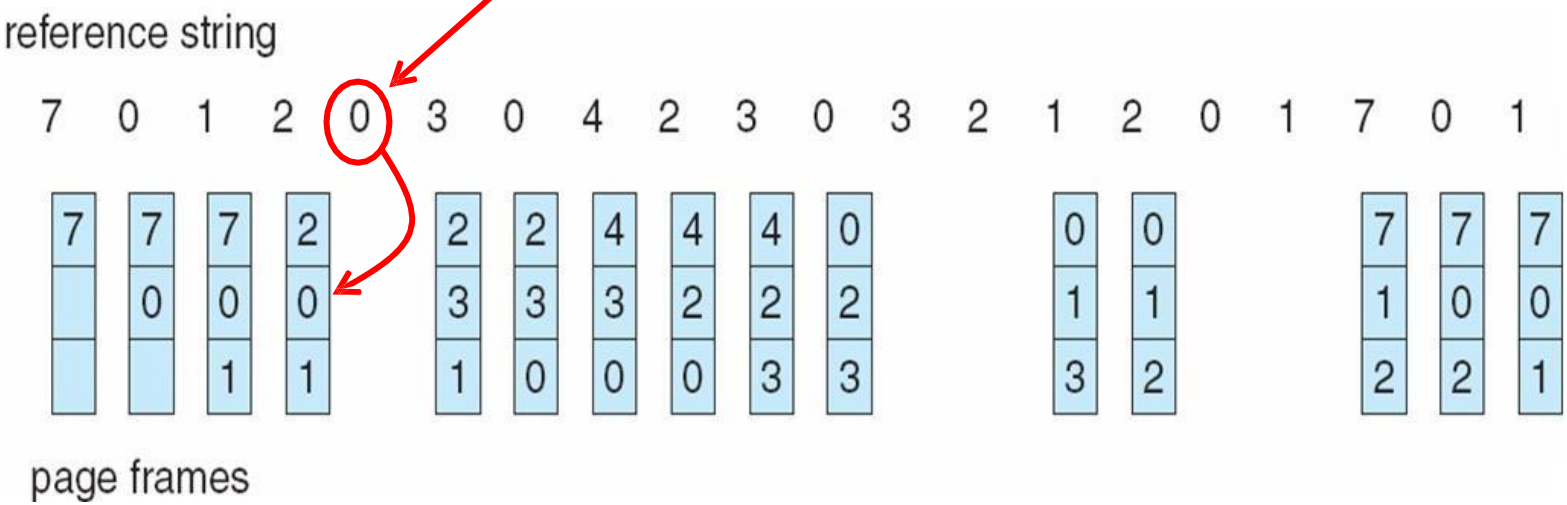
1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

ความผิดปกติของ Belady: เฟรมมากขึ้น ⇒ มีข้อบกพร่องของหน้ามากขึ้น

- Belady's Anomaly: more frames ⇒ more page faults

FIFO Page Replacement

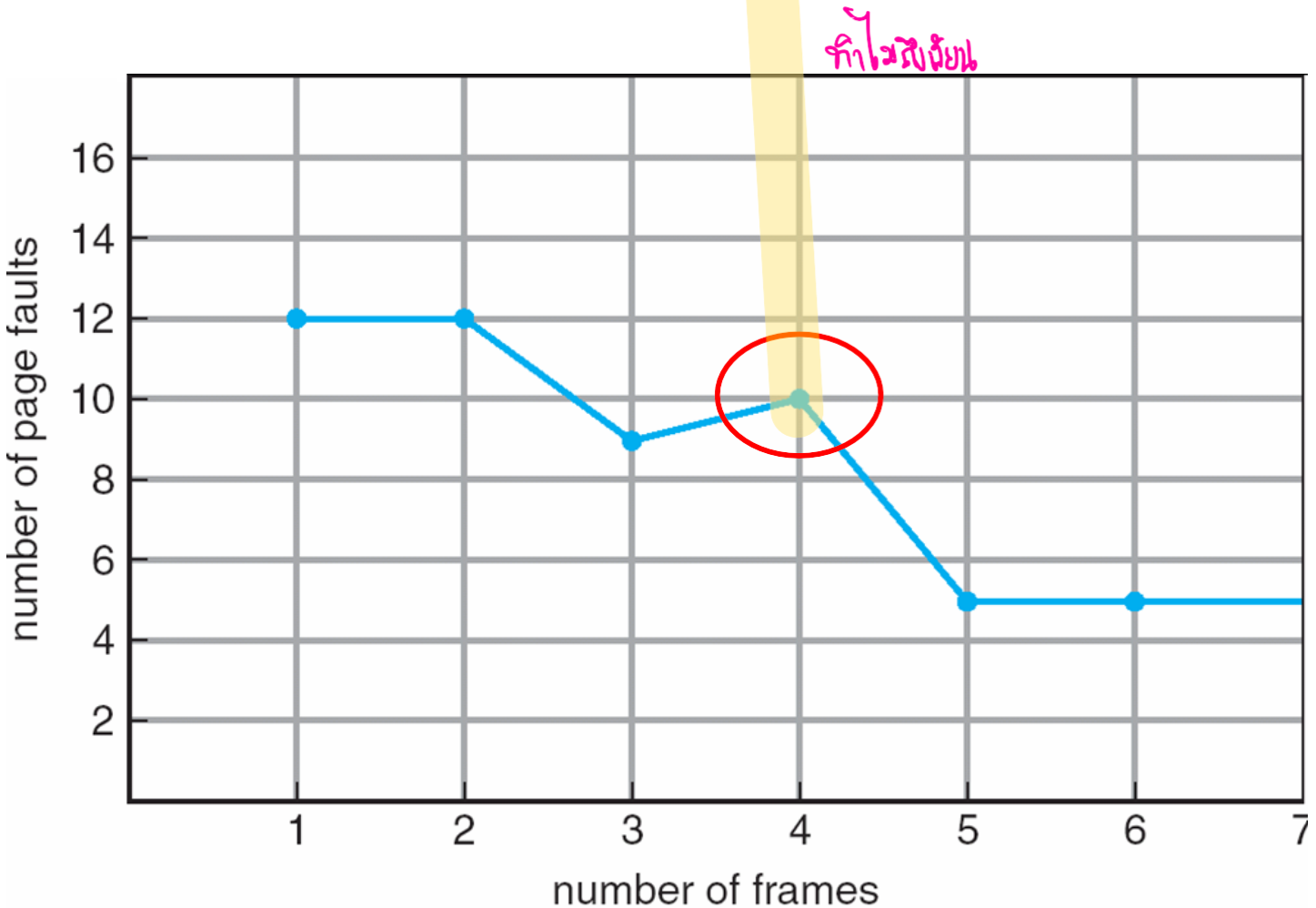
มี 0 อยู่แล้วใน memory จึงไม่ต้องไปดึงข้อ มูลมาใหม่อีก จึงไม่เกิด page fault สำหรับ page 0



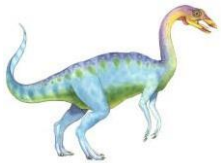
เมื่อ physical memory 3 frames และใช้ reference string ตามที่กำหนดให้ จะเกิด page fault ทั้งหมด 15 page faults

- ครั้งแรกไม่มี page อยู่ใน Physical Memory จะเกิด page fault

FIFO Illustrating Belady's Anomaly



Belady's Anomoly : เป็นเหตุการณ์ที่เมื่อมี Physical Memory เพิ่มขึ้น แต่จะเกิด page fault เพิ่มขึ้นด้วย (เป็นข้อยกเว้นสำหรับ FIFO Algorithm)

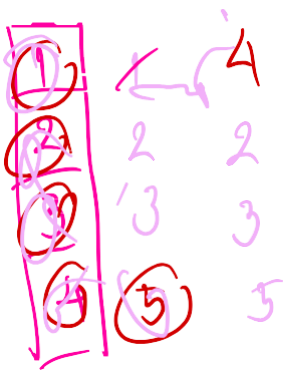


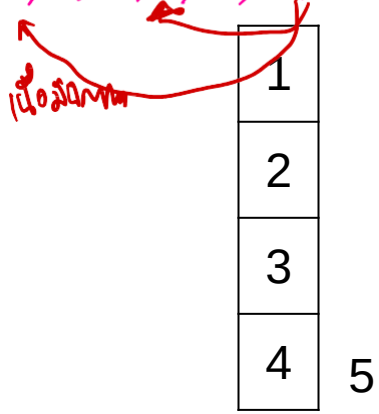
Optimal Algorithm อัลกอริธึมที่เหมาะสมที่สุด

Victim = the farthest on the right

- แทนที่เพจที่จะไม่ถูกใช้งานเป็นระยะเวลานานที่สุด
- Replace page that will not be used for longest period of time
  (**มองใน list ของ reference string ถัดไปในอนาคตว่า page ใดอีกนานที่จะถูกใช้งาน จะโดน replace**)
- 4 frames example

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

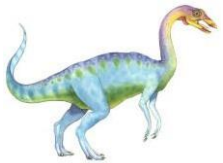




6 page faults

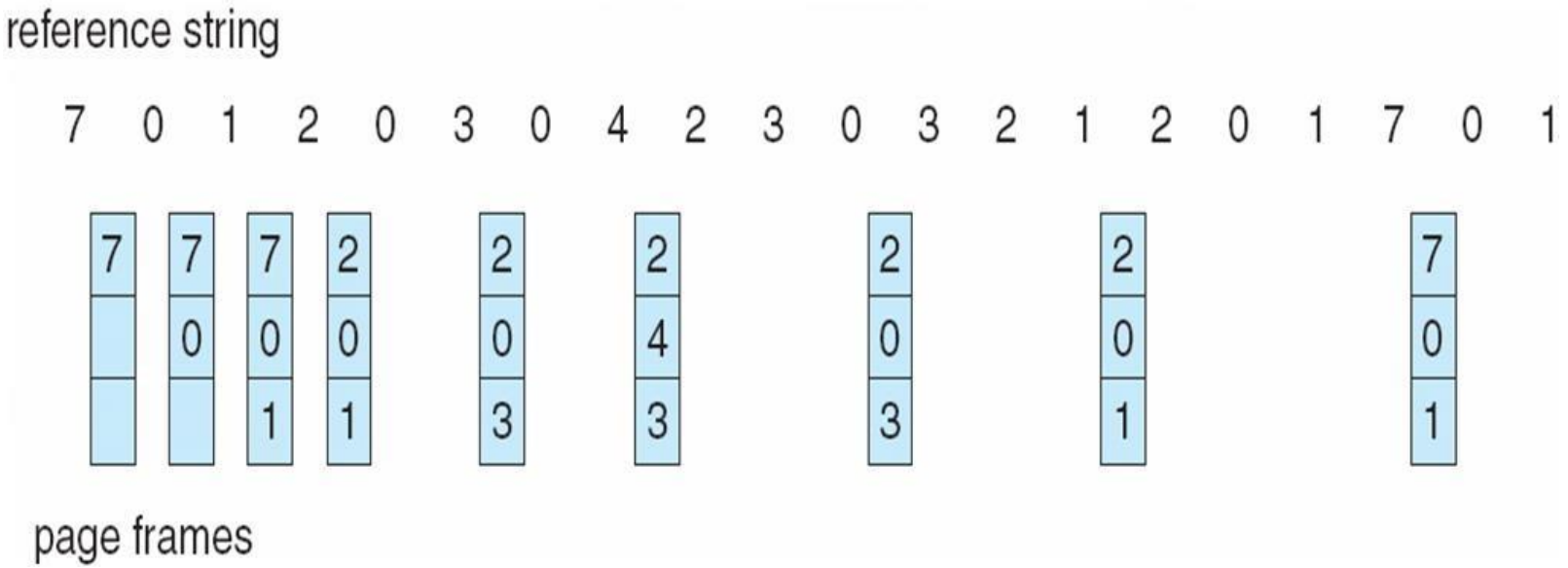
คุณรู้เรื่องนี้ได้อย่างไร?

- How do you know this? **ไม่ make sence เราไม่สามารถรู้อนาคตได้**
- Used for measuring how well your algorithm performs
  ใช้สำหรับวัดว่าอัลกอริทึมของคุณทำงานได้ดีเพียงใด

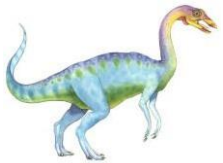


การเปลี่ยนหน้าที่เหมาะสมที่สุด

Optimal Page Replacement

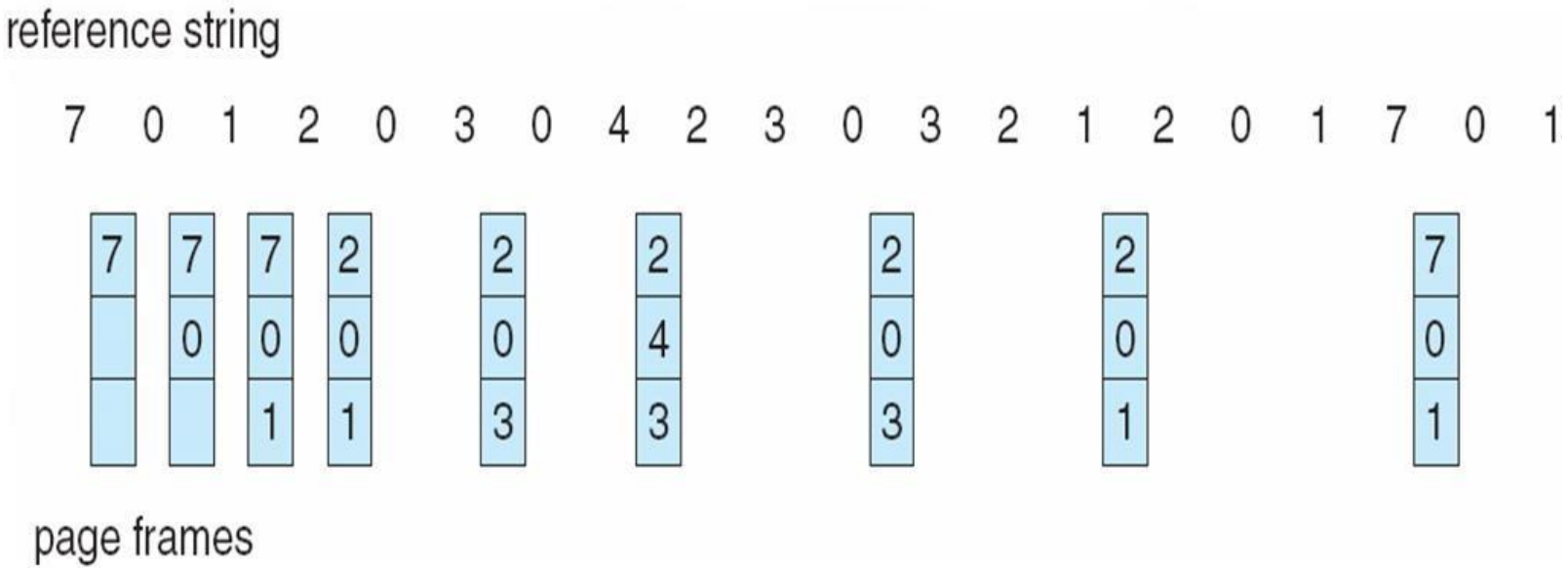


เกิดกี่ page fault ???



การเปลี่ยนหน้าที่เหมาะสมที่สุด

Optimal Page Replacement

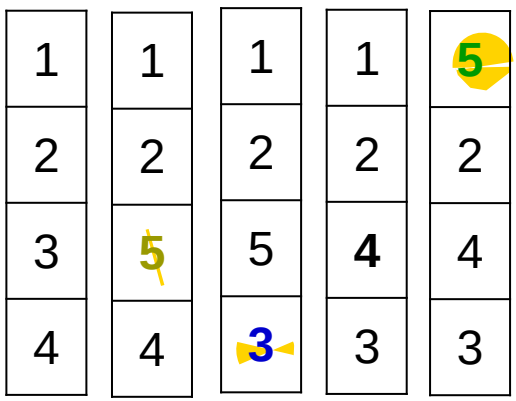


เกิด **9** page fault

Least Recently Used (LRU) Algorithm

(มองย้อนกลับไปในอดีตว่า page ใดไม่ได้ถูกใช้มานานที่สุดจะถูก replace)

- Reference string: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**



การดำเนินการตอบโต้

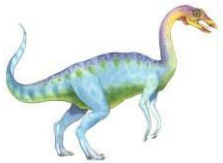
- Counter implementation

ทุกรายการหน้ามีตัวนับ ทุกครั้งที่มีการอ้างอิงหน้าผ่านรายการนี้ ให้คัดลอกนาฬิกาไปที่ตัวนับ

  - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter

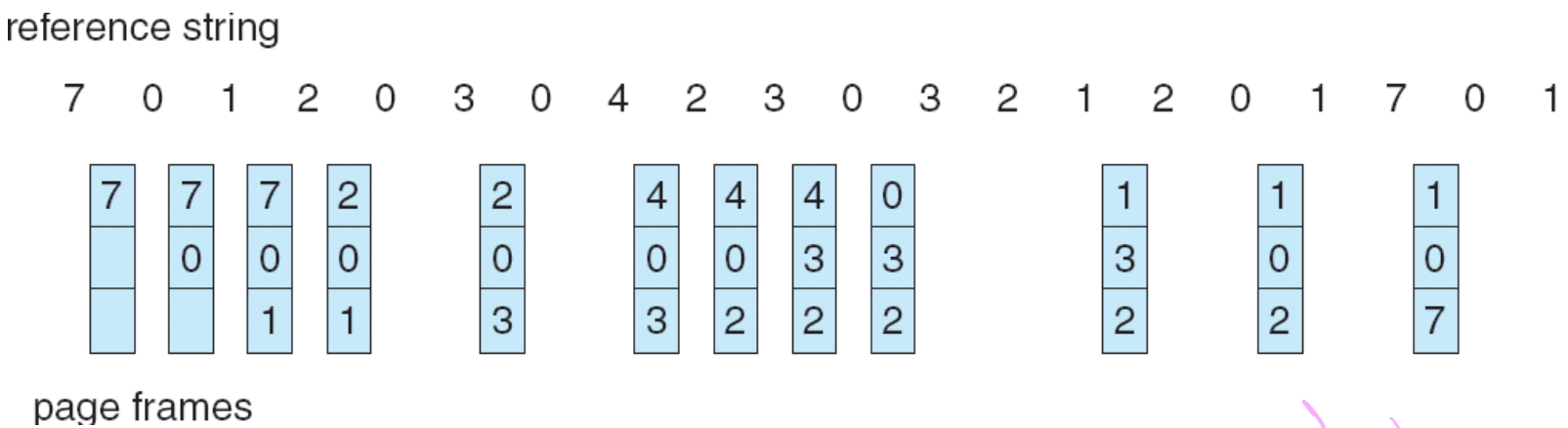
เมื่อจำเป็นต้องเปลี่ยนหน้า ให้ดูที่ตัวนับเพื่อดูว่าหน้าใดที่ต้องเปลี่ยน

  - When a page needs to be changed, look at the counters to determine which are to change

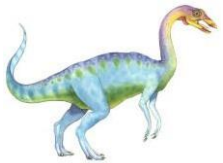


การเปลี่ยนหน้า LRU

LRU Page Replacement



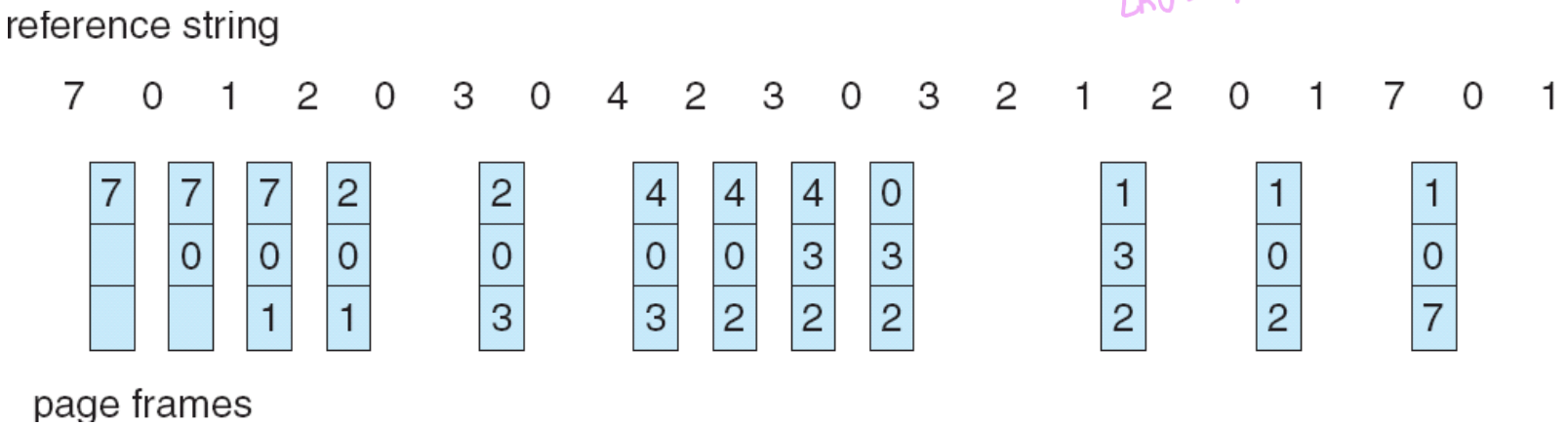
เกิดกี่ page fault ???



การเปลี่ยนหน้า LRU

LRU Page Replacement

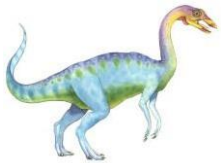
demo 15
optimo 9
LRU = 12



เกิด **12** page fault

ดีกว่า FIFO แต่แย่กว่าอัลกอริทึมที่เหมาะสมที่สุด

- Better than FIFO but worse than Optimal Algorithm



Counting Algorithms อัลกอริทึมการนับ

เก็บตัวนับจำนวนการอ้างอิงที่ทำในแต่ละหน้า

- Keep a counter of the number of references that have been made to each page

(on the left side)
count the frequency of usages (or references) in the past

อัลกอริทึม LFU: แทนที่หน้าที่มีจำนวนน้อยที่สุด

Least frequency

- **LFU Algorithm**: replaces page with smallest count

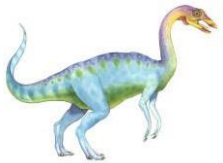
MFU Algorithm: จากการโต้แย้งว่าเพจที่มีการนับน้อยที่สุดอาจเพิ่งนำเข้ามาและยังไม่ได้ใช้งาน

- **MFU Algorithm**: based on the argument that the page with the smallest count was probably just brought in and has yet to be used

เลขที่ถูกเรียกใช้จะเพิ่มขึ้น

See my note.

LFU : Least Frequently Used (ความถี่ในการถูกใช้น้อยที่สุด)
MFU: Most Frequently Used (ความถี่ในการถูกใช้มากที่สุด)



Allocation of Frames

การจัดสรรเฟรม

แต่ละกระบวนการต้องการจำนวนหน้าขั้นต่ำ

- Each process needs *minimum* number of pages

ตัวอย่าง: IBM 370 – 6 หน้าสำหรับจัดการคำสั่ง SS MOVE:

- Example: IBM 370 – 6 pages to handle SS MOVE instruction:
  - คำสั่งมีขนาด 6 ไบต์ อาจขยายได้ 2 หน้า
  - instruction is 6 bytes, might span 2 pages
  - 2 หน้าในการจัดการจาก
  - 2 pages to handle *from*
  - 2 หน้าที่ต้องจัดการ
  - 2 pages to handle *to*

สองแผนการจัดสรรที่สำคัญ

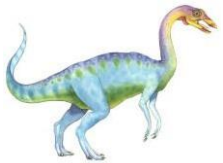
- Two major allocation schemes
  - การจัดสรรคงที่
  - fixed allocation
  - การจัดสรรลำดับความสำคัญ
  - priority allocation

IBM's SS format (substring search)

4 bytes / page

หน้า 34

หน้า 35



Fixed Allocation การจัดสรรคงที่

Method 1 การจัดสรรที่เท่ากัน – ตัวอย่างเช่น หากมี 100 เฟรมและ 5 กระบวนการ ให้แต่ละกระบวนการมี 20 เฟรม

- Equal allocation – For example, if there are 100 frames and 5 processes, give each process 20 frames. (ได้จาก (frame / process) = (100 /5)

Method 2 การจัดสรรตามสัดส่วน – จัดสรรตามขนาดของกระบวนการ

- Proportional allocation – Allocate according to the size of process
  - s_i = size of process p_i
  - $S = \sum s_i$
  - m = total number of frames
  - a_i = allocation for $p_i = \frac{s_i}{S} \times m$

total of 64 frame $m = 64$

$s_i = 10$ ← P_i

$S = s_1 + s_2 = 10 + 127 = 137$ (bytes or page) $s_2 = 127$ ← P_2

P_i ได้ 5 pages frame

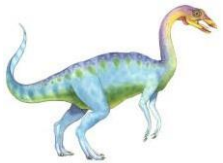
$$a_1 = \frac{10}{137} \times 64 \approx 5$$

$a_1 = (s_1/S) \times m$

P_2 ได้ 59 pages frame

$$a_2 = \frac{127}{137} \times 64 \approx 59$$

$a_2 = (s_2/S) \times m$



Priority Allocation การจัดสรรลำดับความสำคัญ

Fixed allocation uses size for the proportionation allocation
priority allocation uses priority for the proportionation allocation

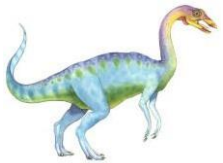
ใช้แผนการจัดสรรตามสัดส่วนโดยใช้ลำดับความสำคัญมากกว่าขนาด

- Use a proportional allocation scheme using priorities rather than size

Higher-priority process (P_i) gets larger allocation (larger a_i) than lower-priority process (P_j) $a_i > a_j$
Thus, the higher-priority process can be executed faster.

- If process P_i generates a page fault, หากกระบวนการ Pi สร้างเพจฟอลต์
  - เลือกเพื่อแทนที่เฟรมใดเฟรมหนึ่ง select for replacement one of its frames
  - เลือกเพื่อแทนที่เฟรมจากกระบวนการที่มีหมายเลขลำดับความสำคัญต่ำกว่า select for replacement a frame from a process with lower priority number

the lower-priority process will be the victim to be replaced first so starvation may occur



การจัดสรรทั่วโลกเทียบกับท้องถิ่น

Global vs. Local Allocation

No free frame is left There must be a replament

การทดแทนทั่วโลก – กระบวนการเลือกเฟรมทดแทนจากชุดของเฟรมทั้งหมด กระบวนการหนึ่งสามารถนำเฟรมมาจากที่อื่นได้

- **Global replacement** – process selects a replacement frame from the set of all frames; one process can take a frame from another

การแทนที่เฉพาะที่ – แต่ละกระบวนการจะเลือกจากชุดเฟรมที่จัดสรรของตัวเองเท่านั้น

- **Local replacement** – each process selects from only its own set of allocated frames

Select some frame of the process's own.

17.5 .
3.75) 22.5 / คนละ
1.25 28

End of Chapter 8