OPERATING SYSTEMS

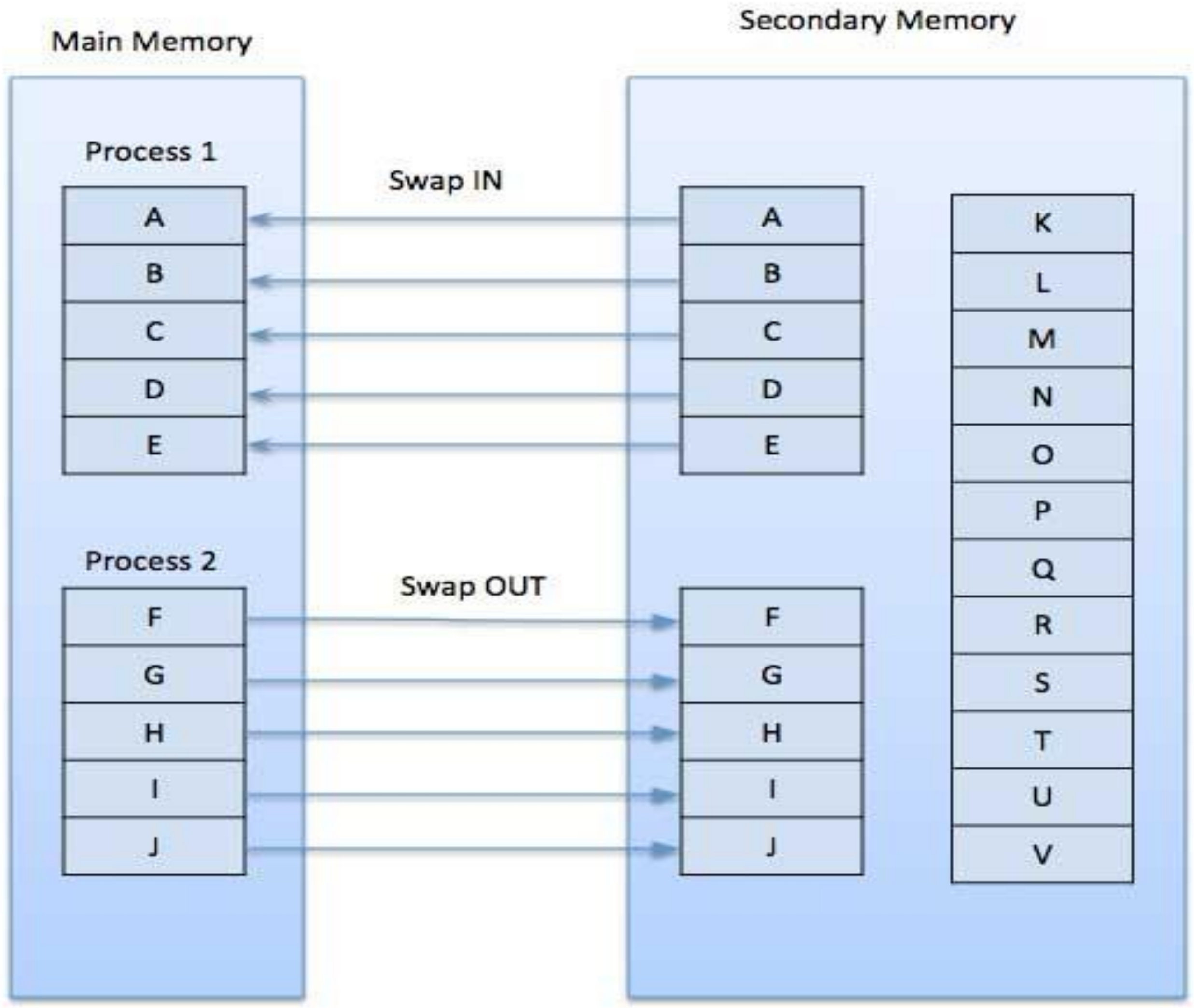
Memory Management

Class Presentations on Operating System

by Subhash Chand Agrawal

Demand Paging

- A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.
- When a context switch occurs, the operating system does not copy any of the old program’s pages out to the disk or any of the new program’s pages into the main memory.
- Instead, it just begins executing the new program after loading the first page and fetches that program’s pages as they are referenced.



https://www.tutorialspoint.com/operating_system/os_virtual_memory.htm

Page Fault

- While executing a program, if the program references a page which is not available in the main memory.
- Processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

Advantages

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

Paging Revisitted

- If a page is not in physical memory
  - find the page on disk
  - find a free frame
  - bring the page into memory
- What if there is no free frame in memory?

Page Replacement

- Basic idea
  - if there is a free page in memory, use it
  - if not, select a *victim* frame
  - write the victim out to disk
  - read the desired page into the now free frame
  - update page tables
  - restart the process

Page Replacement

- Main objective of a good replacement algorithm is to achieve a low *page fault rate*
  - insure that heavily used pages stay in memory
  - the replaced page should not be needed for some time
- Secondary objective is to reduce latency of a page fault
  - efficient code
  - replace pages that do not need to be written out

Reference String

- Reference string is the sequence of pages being referenced
- If user has the following sequence of addresses
  - 123, 215, 600, 1234, 76, 96
- If the page size is 100, then the reference string is
  - 1, 2, 6, 12, 0, 0

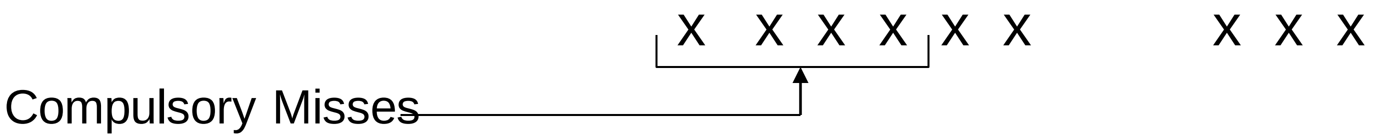
First In First Out (FIFO) algorithm

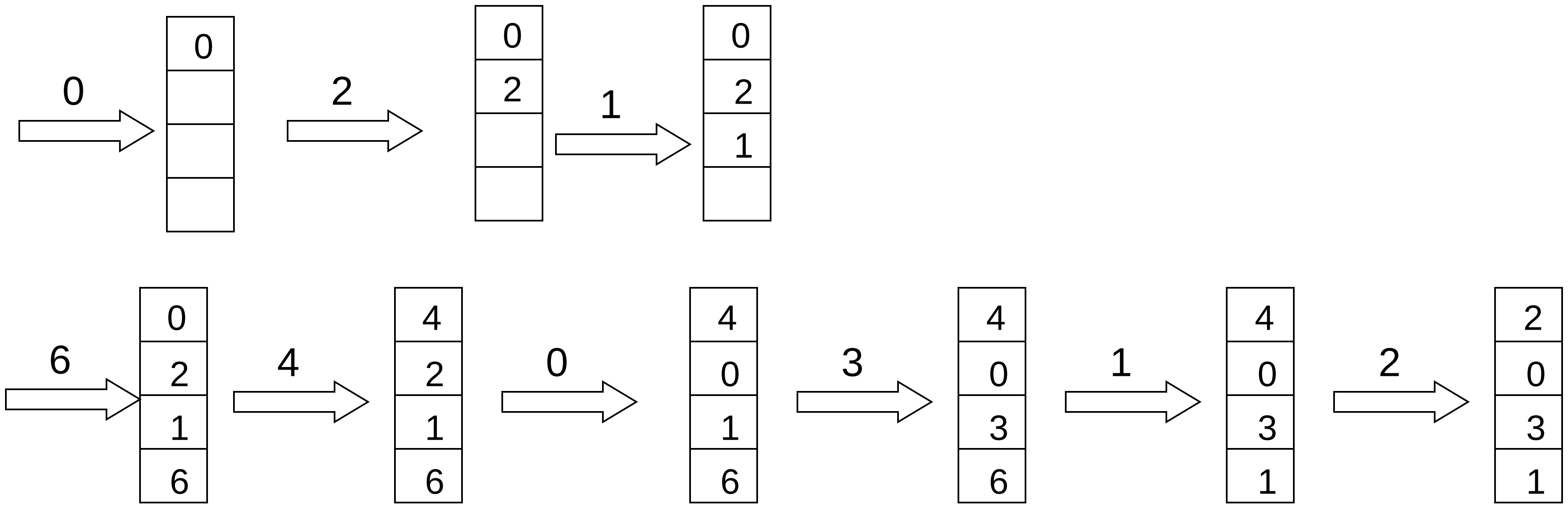
- Oldest page in main memory is the one which will be selected for replacement.
- Very simple to implement
  - keep a list
    - victims are chosen from the tail
    - new pages in are placed at the head

FIFO

- Fault Rate = 9 / 12 = 0.75

- Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1





FIFO Issues

- Poor replacement policy
- Evicts the oldest page in the system
  - usually a heavily used variable should be around for a long time
  - FIFO replaces the oldest page - perhaps the one with the heavily used variable
- FIFO does not consider page usage

- Giving more memory to process would improve its performance?

- Consider the following reference string:

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Calculate the page fault rate for 3 frames and 4 frames with FIFO page replacement algorithm.

Belady’s Anomaly:

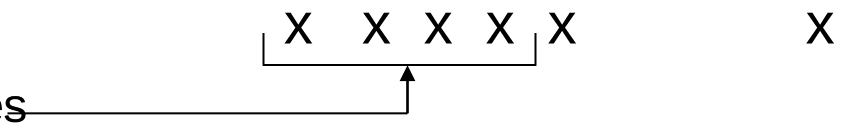
- For some page replacement algorithms, the page fault rate may increase as the number of allocated frames increases.

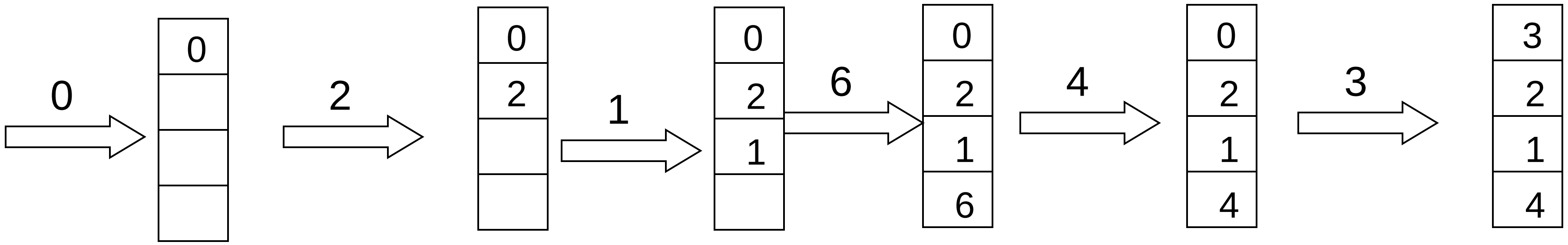
Optimal Page Replacement

- Basic idea
  - replace the page that will not be referenced for the longest time
- This gives the lowest possible fault rate

Optimal Page Replacement(Future)

•Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1





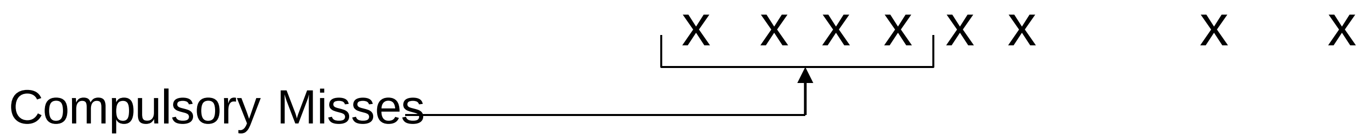
•Fault Rate = 6 / 12 = 0.50
•With the above reference string, this is the best we can hope to do

Least Recently Used (LRU)(Past)

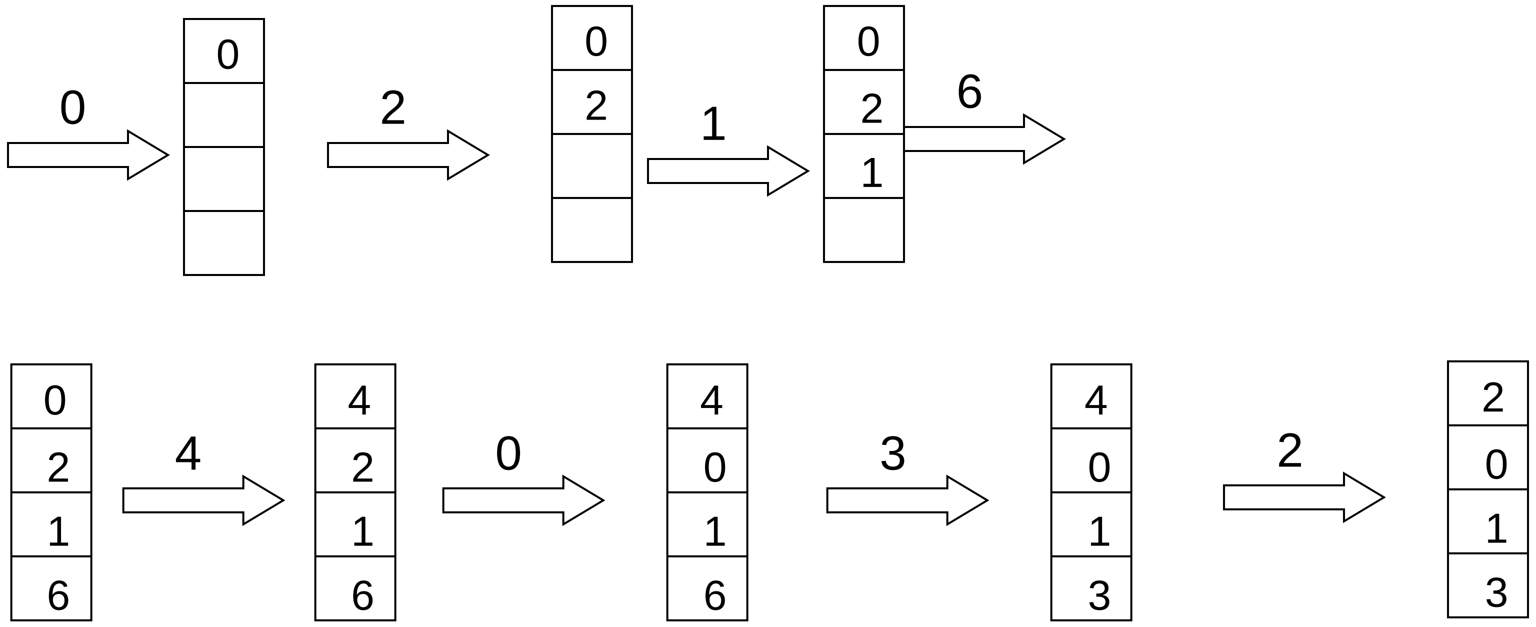
- Basic idea
  - replace the page in memory that has not been accessed for the longest time
- Optimal policy looking back in time
  - as opposed to forward in time
  - fortunately, programs tend to follow similar behavior

LRU

- Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1



- Fault Rate = 8 / 12 = 0.67



LRU Issues

- How to keep track of last page access?
  - requires special hardware support
- 2 major solutions
  - Counters
    - the page with the smallest “time” value is replaced
  - stack
    - keep a stack of references
    - on every reference to a page, move it to top of stack
    - page at bottom of stack is next one to be replaced

Q1

- Find the page fault rate by considering the following reference string:
- 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 using three frames with
- (i) FIFO
- (ii) Optimal
- (iii) LRU

(i)15
(ii) 9
(iii) 12

Q2

- Consider the following page reference string: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

  How many page faults would occur for replacement by LRU, FIFO, optimal, for one, two, three, four, five, six and seven frames? All frames are initially empty and first unique page reference causes a page fault.

# frames	1	2	3	4	5	6	7
LRU	20	18	15	10	8	7	7
FIFO	20	18	16	14	10	10	7
Optimal	20	15	11	8	7	7	7

- In a demand-paged system, the degree of multiprogramming is fixed at 4. Measurements were made to determine the utilization of cpu and the paging disk. The result is one of the following. For each of case a, b, c, state in one line what is happening?
- a. cpu utilization 13%, disk 97%
- b. cpu utilization 92%, disk 11%
- c. cpu utilization 21%, disk 3%

- Answer:
- a. Thrashing
- b. okay
- c. can increase multiprogramming