

CSC 4103 - Operating Systems
Fall 2009

LECTURE - XIX
VIRTUAL MEMORY - II

Tevfik Koşar

Louisiana State University
November 3rd, 2009

Least Recently Used (LRU) Algorithm

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

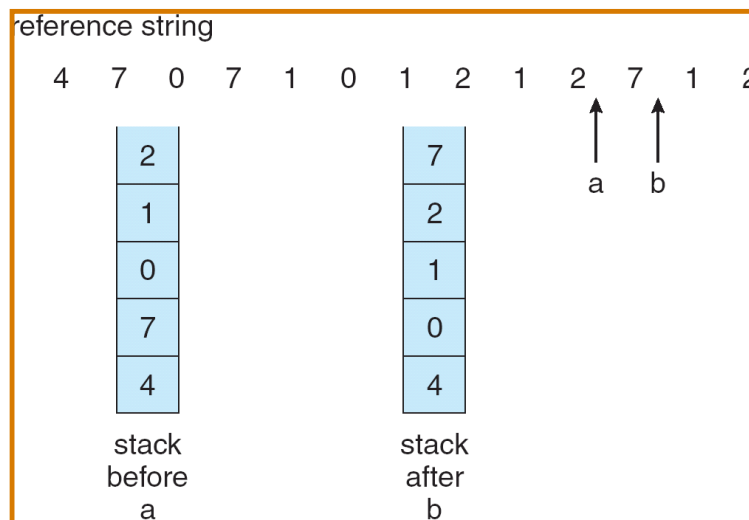
1	5
2	
3	5 4
4	3

- How to implement??

LRU Algorithm (Cont.)

- **Counter implementation** (Needs hardware assistance)
 - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
 - When a page needs to be changed, look at the counters to determine which are to change
- **Stack implementation** - keep a stack of page numbers in a double link form:
 - Page referenced:
 - move it to the top
 - requires 6 pointers to be changed
 - No search for replacement

Use Of A Stack to Record The Most Recent Page References



LRU Approximation Algorithms

- **Reference bit**

- With each page associate a bit, initially = 0
- When page is referenced bit set to 1
- Replace the one which is 0 (if one exists). We do not know the order, however.

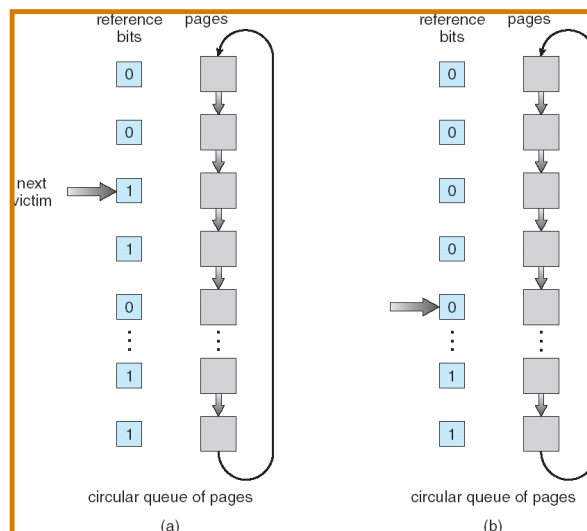
- **Additional Reference bits**

- 1 byte for each page: eg. 00110011
- Shift right at each time interval

- **Second chance**

- Need reference bit
- Clock replacement
- If page to be replaced (in clock order) has reference bit = 1 then:
 - set reference bit 0
 - leave page in memory
 - replace next page (in clock order), subject to same rules

Second-Chance (clock) Page-Replacement Algorithm



Counting Algorithms

- Keep a counter of the number of references that have been made to each page
- **LFU Algorithm:** replaces page with smallest count
- **MFU Algorithm:** based on the argument that the page with the smallest count was probably just brought in and has yet to be used

Allocation of Frames

- Each process needs *minimum* number of pages
- Two major allocation schemes
 - fixed allocation
 - priority allocation

Fixed Allocation

- **Equal allocation** - For example, if there are 100 frames and 5 processes, give each process 20 frames.
- **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 $p_i = \frac{s_i}{S} \times m$

$m = 64$

$s_1 = 10$

$s_2 = 127$

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

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

Priority Allocation

- Use a proportional allocation scheme using priorities rather than size
- If process P_i generates a page fault,
 - select for replacement one of its frames
 - select for replacement a frame from a process with lower priority number

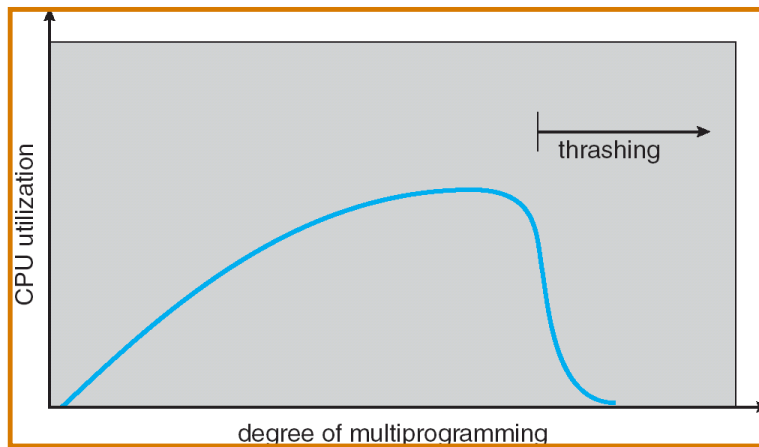
Global vs. Local Allocation

- **Global replacement** - process selects a replacement frame from the set of all frames; one process can take a frame from another
- **Local replacement** - each process selects from only its own set of allocated frames

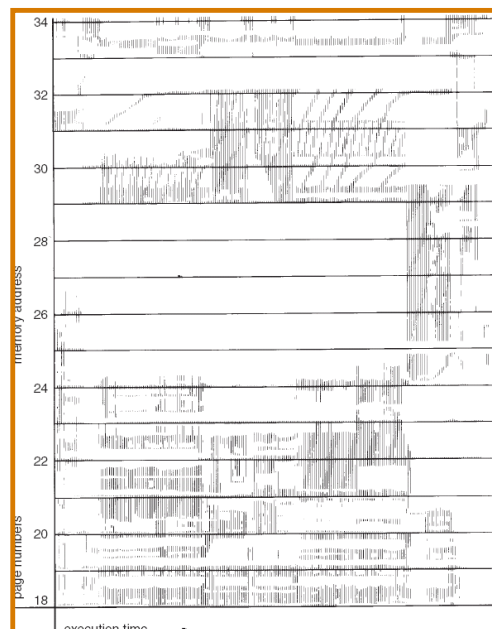
Thrashing

- If a process does not have “enough” frames, the page-fault rate is very high. This leads to:
 - Replacement of active pages which will be needed soon again
 - **Thrashing** = a process is busy swapping pages in and out
- Which will in turn cause:
 - low CPU utilization
 - operating system thinks that it needs to increase the degree of multiprogramming
 - another process added to the system

Thrashing (Cont.)



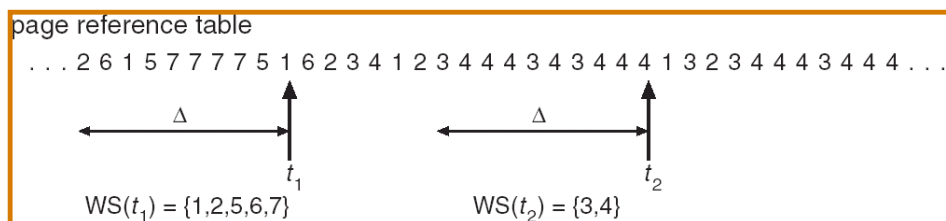
Locality in a Memory-Reference Pattern



Working-Set Model

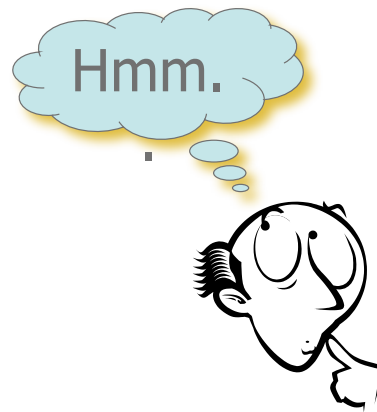
- $\Delta \equiv$ working-set window \equiv a fixed number of page references
Example: 10,000 instruction
- WSS_i (working set of Process P_i) =
total number of pages referenced in the most recent Δ (varies in time)
 - if Δ too small will not encompass entire locality
 - if Δ too large will encompass several localities
 - if $\Delta = \infty \Rightarrow$ will encompass entire program
- $D = \sum WSS_i \equiv$ total demand frames
- if $D > m \Rightarrow$ Thrashing
- Policy if $D > m$, then suspend one of the processes

Working-set model



Summary

- Virtual Memory
 - Demand Paging
 - Page Faults
 - Page Replacement
 - Page Replacement Algorithms
 - (FIFO, LRU, Optimal etc)
 - Performance of Demand Paging



- **Reading Assignment: Chapter 8 from Silberschatz.**

Acknowledgements

- “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings
- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR