

BUCKNELL UNIVERSITY
Computer Science

CSCI 315 Operating Systems Design

Allocation of Frames & Thrashing (Virtual Memory)

04/02/2010

CSCI 315 Operating Systems Design

1

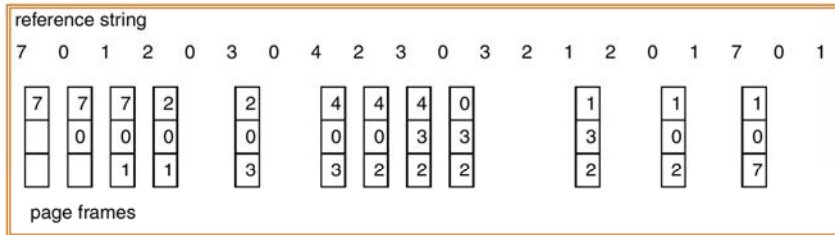
LRU Algorithm

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

1	5
2	
3	5 4
4	3

- Counter implementation:
 - 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.

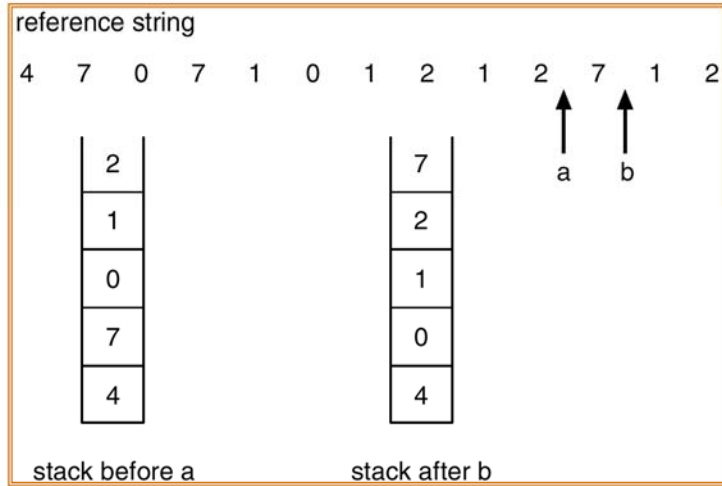
LRU Page Replacement



LRU Algorithm (Cont.)

- 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



04/02/2010

CSCI 315 Operating Systems Design

5

LRU and Belady's Anomaly

- LRU does not suffer from Belady's Anomaly (OPT doesn't either).
- It has been shown that algorithms in a class called **stack algorithms** can never exhibit Belady's Anomaly.
- A **stack algorithm** is one for which the set of pages in memory for n frames is a subset of the pages that could be in memory for $n+1$ frames.

LRU Approximation Algorithms

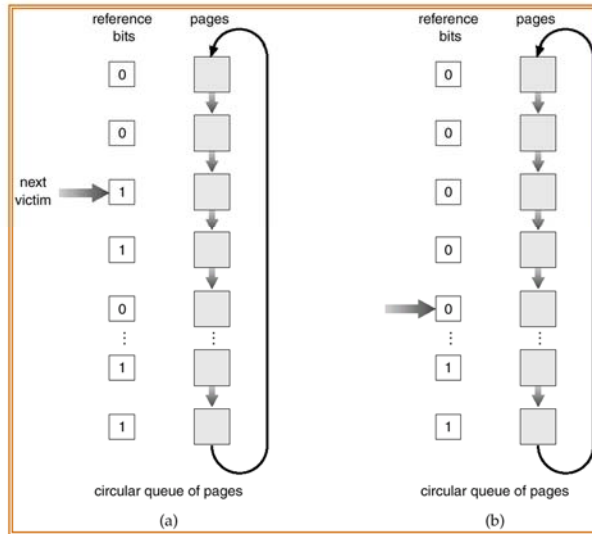
- **Reference bit**

- 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.

- **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



04/02/2010

CSCI 315 Operating Systems Design

8

Counting Algorithms

- Keep a counter of the number of references that have been made to each page.
- **LFU Algorithm:** Replaces page with smallest count. The counters should be “aged”: pages that are referenced many times but only for a small period of time would hang around otherwise.
- **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 a **minimum** number of pages.
- There are two major allocation schemes:
 - **fixed allocation**
 - **priority allocation**

Fixed Allocation

- **Equal allocation** – e.g., if 100 frames and 5 processes, give each 20 pages.
- **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 $p_i = \frac{s_i}{S} \times m$

$$\left\{ \begin{array}{l} m = 64 \\ s_1 = 10 \\ s_2 = 127 \\ a_1 = \frac{10}{137} \times 64 \approx 5 \\ a_2 = \frac{127}{137} \times 64 \approx 59 \end{array} \right.$$

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.

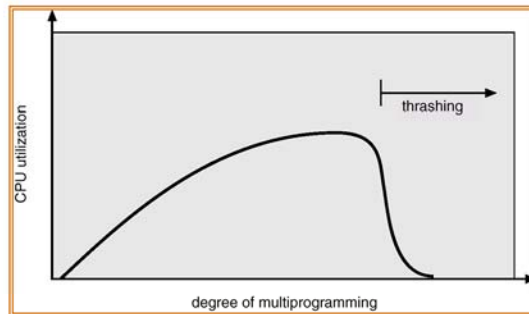
Global vs. Local Allocation

- **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

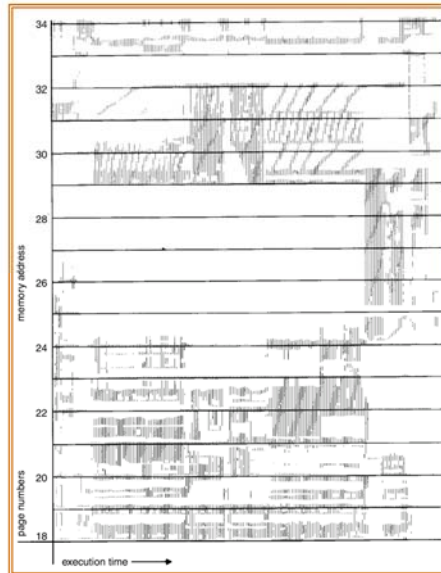
- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
 - **Low CPU utilization.**
 - Operating system thinks that it needs to increase the degree of multiprogramming.
 - Another process added to the system.
- **Thrashing** \equiv a process is busy swapping pages in and out.

Thrashing



- Why does **paging** work?
Locality model
 - Process migrates from one locality to another.
 - Localities may overlap.
- Why does **thrashing** occur?
 Σ size of locality > total memory size

Locality in Memory-Reference Pattern




04/02/2010

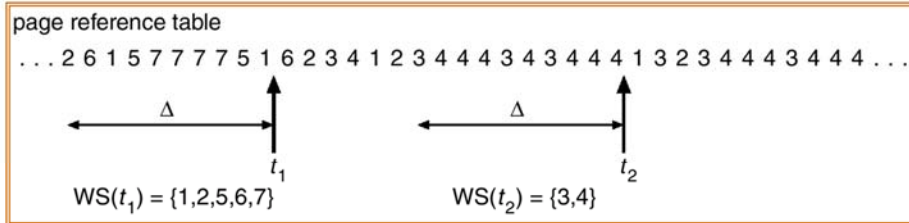
CSCI 315 Operating Systems Design

16

Working-Set Model

- $\Delta \equiv$ **working-set window** \equiv a fixed number of page references.
- 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.

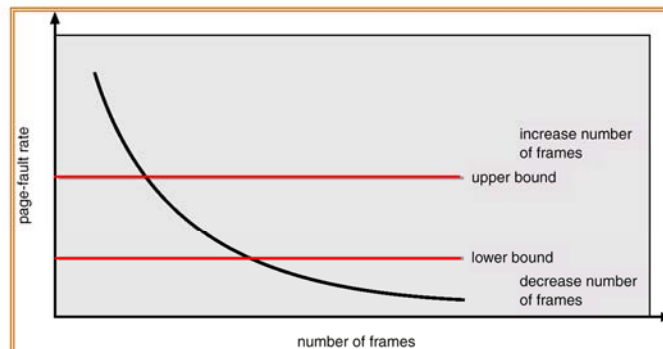
Working-set model



Keeping Track of the Working Set

- Approximate with interval timer + a reference bit
- Example: $\Delta = 10,000$
 - Timer interrupts after every 5000 time units.
 - Keep in memory 2 bits for each page.
 - Whenever a timer interrupts copy and sets the values of all reference bits to 0.
 - If one of the bits in memory = 1 \Rightarrow page in working set.
- Why is this not completely accurate?
- Improvement = 10 bits and interrupt every 1000 time units.

Page-Fault Frequency Scheme



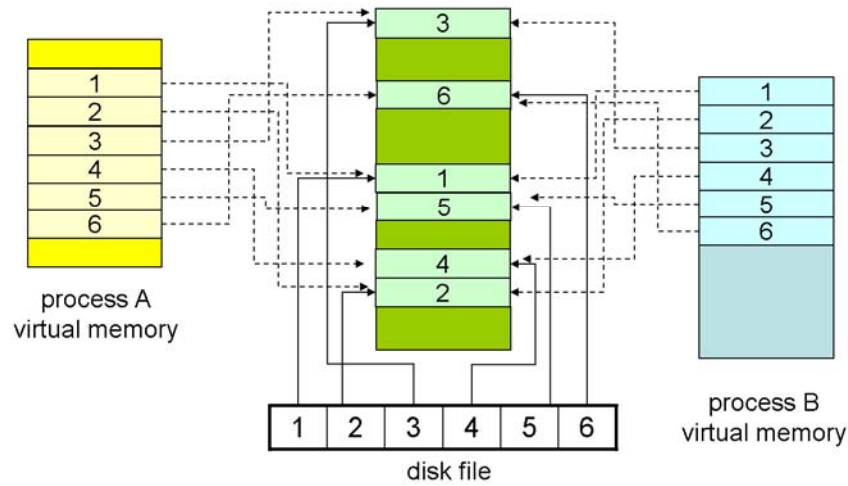
Establish "acceptable" page-fault rate.

- If actual rate too low, process loses frame.
- If actual rate too high, process gains frame.

Memory-mapped Files

- Memory mapping a file can be accomplished by mapping a disk block to one or more pages in memory.
- A page-sized portion of the file is read from the file system into a physical page. Subsequent `read()` and `write()` operations are handled as memory (not disk) accesses.
- Writing to the file in memory is not necessarily synchronous to the file on disk. The file can be committed back to disk when it's closed.

Memory-mapped Files



04/02/2010

CSCI 315 Operating Systems Design

22

Prepaging

- **Prepaging:** In order to avoid the initial number of page faults, the system can bring into memory all the pages that will be needed all at once.
- This can also be applied when a swapped-out process is restarted. The smart thing to do is to remember the working set of the process.
- One question that arises is whether all the pages brought in will actually be used...
- Is the cost of prepaging less than the cost of servicing each individual page fault?