Introduction to OS

Page Replacement Algorithm

MOS 3.4

Mahmoud El-Gayyar

elgayyar@ci.suez.edu.eg

Review

- *Virtual Memory* — the address space in which a process "thinks"
  - As distinguished from *Physical Memory*, the address space of the hardware memory system
- Multiple forms
  - *Base* and *Limit* registers
  - *Segmentation*
  - *Paging*
- *Memory Management Unit* (MMU)
  - Present in most modern processors
  - Converts **all** *virtual addresses* to *physical addresses*
  - Transparent to execution of (almost all) programs

PTE Structure

1	1	1	3	20
V	R	M	prot	page frame number

- *Valid* bit gives state of this PTE
  - says whether or not a virtual address is valid – in memory and VA range
  - If not set, page might not be in memory or *may not even exist!*
- *Reference* bit says whether the page has been accessed
  - it is set by hardware *whenever* a page has been read or written to
- *Modify* bit says whether or not the page is *dirty*
  - it is set by hardware during *every* write to the page
- *Protection* bits control which operations are allowed
  - read, write, execute, etc.
- *Page* frame number (PFN) determines the physical page
  - physical page start address

Useful terms

- ***Thrashing***
  - Too many page faults per unit time
  - Results from insufficient physical memory to support the working set
  - System spends most of its time swapping pages in and out, rather than executing process
- ***Working set***
  - The set of pages needed to keep a process from thrashing
- ***Caching***
  - The art and science of keeping the most active elements in fast storage for better execution speed
  - Depends upon *locality* of references

VM Page Replacement

- If there is an unused frame, use it.

- If there are no unused frames available, select a *victim* (according to policy) and
  - If it contains a dirty page (M == 1)
    - write it to disk
  - Load in new page from disk (or create new page)
  - Restart the faulting instruction

- What is cost of replacing a page?
- How does the OS select the page to be evicted?

Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- *Reference string* – ordered list of pages accessed as process executes

`Ex. Reference String is A B C A B D A D B C B`

The Best Page to Replace

- The best page to replace is the one that will ***never*** be accessed again
- **Optimal Algorithm**
  - Lowest fault rate for any reference string
  - Basically, replace the page *that will not be used for the longest time in the future*.
  - Not realistic !!
  - We want to find close approximations: Use it as an estimation.

Page Replacement – NRU (*N*ot *R*ecently *U*sed)

- Periodically (e.g., on a clock interrupt)
  - Clear R bit from all PTE’s
- When needed, rank order pages as follows
  1. R = 0, M = 0
  2. R = 0, M = 1
  3. R = 1, M = 0
  4. R = 1, M = 1
- Evict a page at random from lowest non-empty class (write out if M = 1)
- Characteristics
  - Easy to understand and implement
  - Not optimal, but adequate in some cases

Page Replacement – FIFO
(***F***irst ***I***n, ***F***irst ***O***ut)

- Easy to implement
  - When swapping a page in, place its *page id* on end of list
  - Evict page at head of list

- Page to be evicted has been in memory the longest time, but …
  - Maybe it is being used, very active even
  - We just don’t know

- FIFO is rarely used in practice

Second Chance

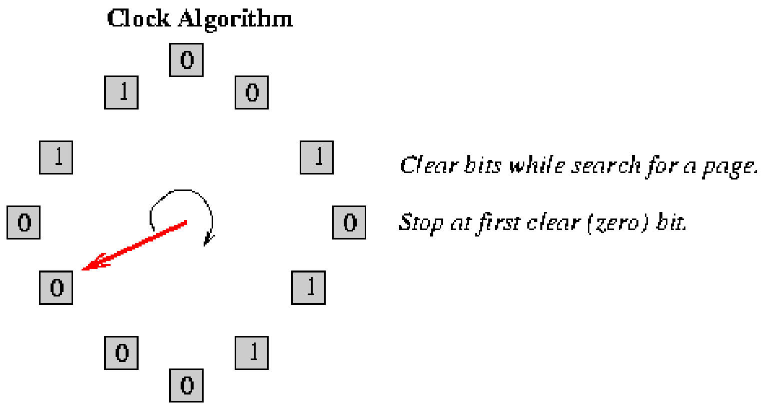
- Maintain FIFO page list
- When a page frame is needed, check reference bit of top page in list
    - If R == 1 then move page to end of list and clear R, repeat
    - If R == 0 then evict page
- I.e., a page has to move to top of list at least twice
    - I.e., once *after* the last time R-bit was cleared
- Disadvantage
    - Moves pages around on list a lot (too much overhead)

Clock Replacement
(A slight variation of Second Chance)

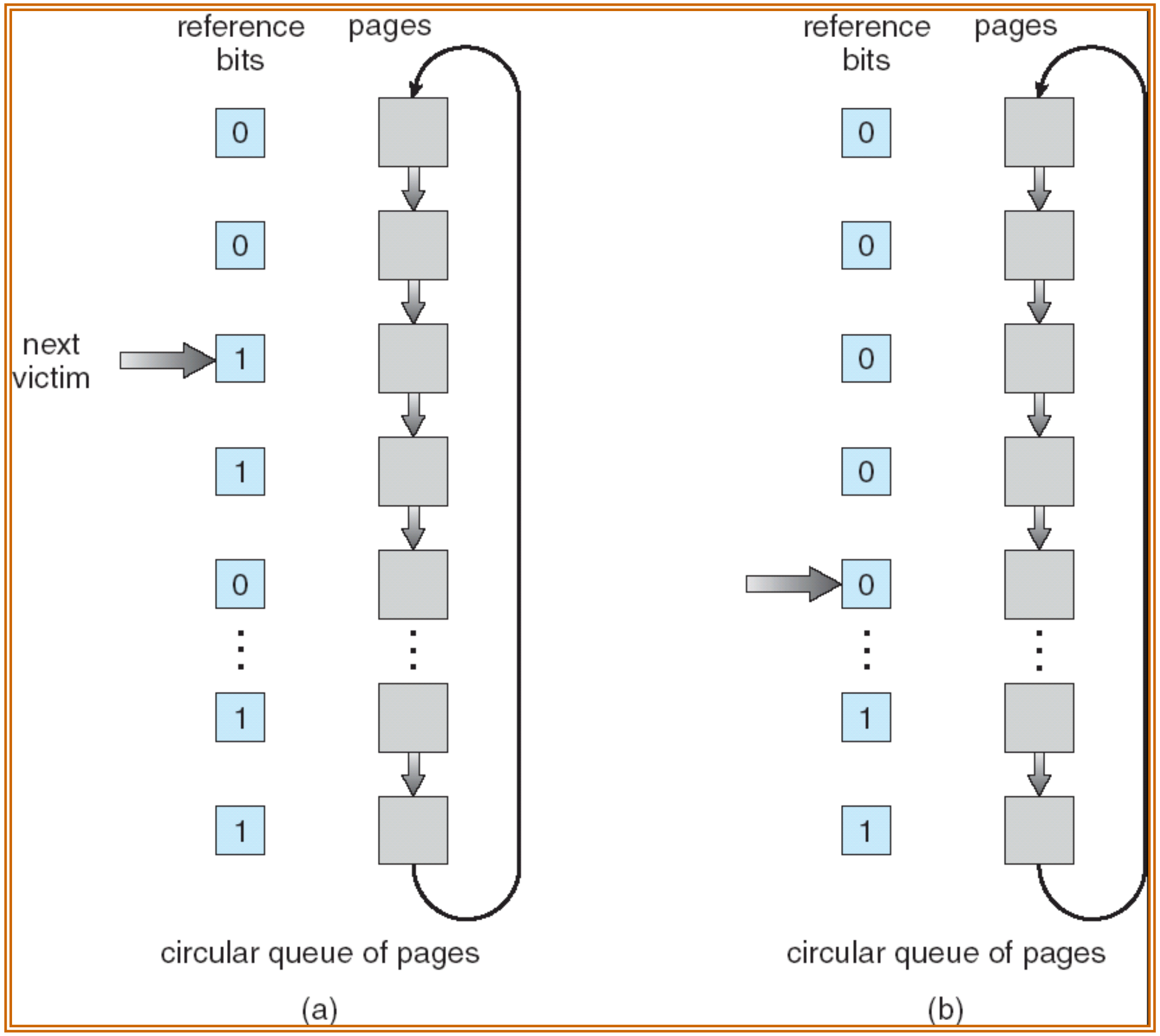
- Create circular list of PTEs in FIFO Order

- One-handed *Clock* – pointer starts at oldest page
  - Algorithm – FIFO, but check Reference bit
    - If R == 1, set R = 0 and advance hand
    - evict first page with R == 0

- Fast, but worst case may take a lot of time

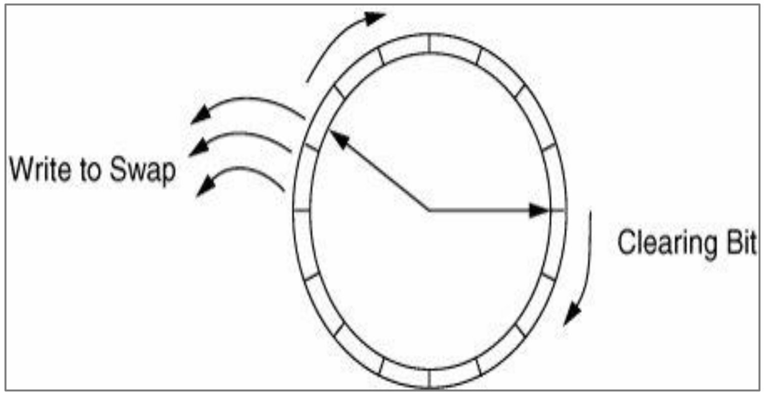

Clock Algorithm
0
1
0
1
1
0
0
0
1
0
1
0
Clear bits while search for a page.
Stop at first clear (zero) bit.

Clock Algorithm (illustrated)



Enhanced Clock Algorithm

- Two-handed *clock* – add another hand that is *n* PTEs ahead
  - Extra hand clears Reference bit
  - Allows very active pages to stay in longer
- Also rank order the frames
  1. R = 0, M = 0
  2. R = 0, M = 1
  3. R = 1, M = 0
  4. R = 1, M = 1



Select first entry in lowest category

- May require multiple passes
- Gives preference to modified pages

Least Recently Used (LRU)

- Replace the page that has not been used for the longest time

3 Page Frames   Reference String - A B C A B D A D B C

LRU – 5 faults

A B C A B D A D B C

LRU

- Past experience may indicate future behavior
- Perfect LRU requires some form of timestamp to be associated with a PTE on every memory reference !!!
- Counter implementation
  - Every page entry has a counter; each 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 to select
- Stack implementation – keep a stack of page numbers in a double link form:
  - Page referenced: move it to the top
  - No search for replacement but movement is expensive

LRU Approximations - **Aging**

- Aging
  - – Keep a counter for each PTE
  - – Periodically (clock interrupt) – check R-bit
    - If R = 0 increment counter (page has not been used)
    - If R = 1 clear the counter (page has been used)
    - Clear R = 0
  - – Counter contains # of intervals since last access
  - – Replace page having largest counter value

Working Set – I

- A *working set* of a process is used to model the dynamic locality of its memory usage
  - *Working set* = set of pages a process currently needs to execute without too many page faults

- Definition:
  - WS(t,w) = set of pages referenced in the interval between time t-w and time t
    - t is time and w is working set window (measured in memory refs)
    - Page is in working set only if it was referenced in last w memory references

Working Set - II

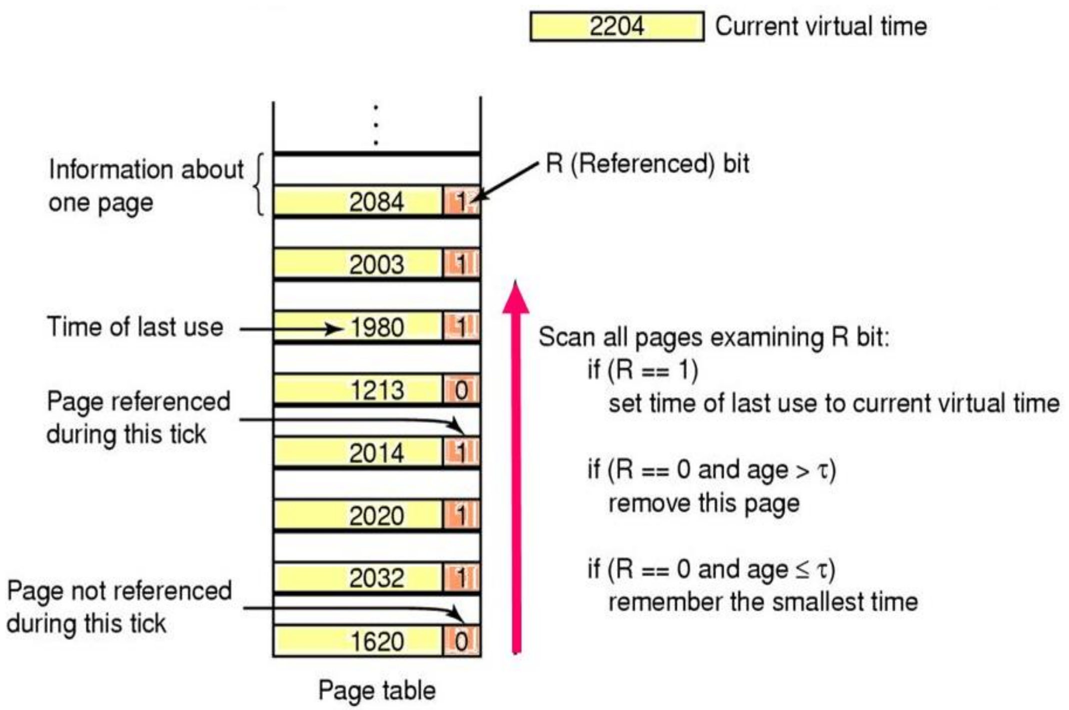
- $w \equiv$ working-set window $\equiv$ a fixed number of page (memory) references
  Example: 10,000 – 2,000,000 instructions
- WS_i (working set of Process P_i) =
  set of pages referenced in the most recent w (varies in time)
  - if w too small will not encompass entire locality.
  - if w too large will encompass several localities.
  - as $w \Rightarrow \infty$, encompasses entire program.

Working Set Page Replacement

- In practice, convert references into time
  - E.g. 100ns/ref, 100,000 references ≅ 10msec
- WS algorithm in practice
  - On each clock tick, clear all R bits and record ***process virtual time*** *t*
  - When looking for eviction candidates, scan all pages of process in physical memory
    - If R == 1

      Store *t* in *LTU* (last time used) of PTE and clear R
    - If R == 0

      If (*t* – *LTU*) > *WS_Interval* (i.e., *w*), evict the page (because it is not in working set)
    - Else select page with the largest difference

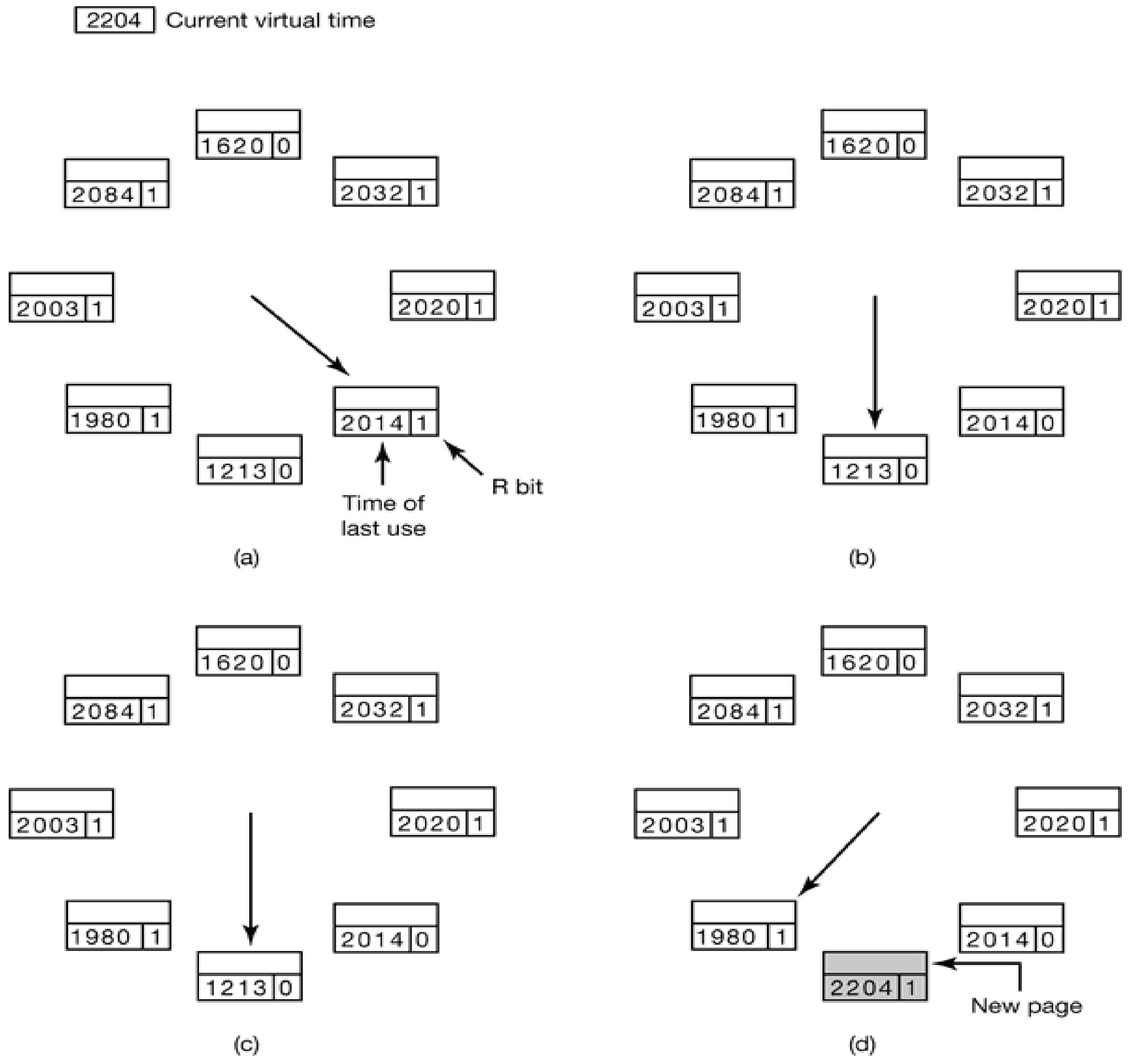


2204
Current virtual time
Information about one page
R (Referenced) bit
2084
1
2003
1
Time of last use
1980
1
Page referenced during this tick
1213
0
2014
1
2020
1
Page not referenced during this tick
2032
1
1620
0
Page table
Scan all pages examining R bit:
if (R == 1)
set time of last use to current virtual time
if (R == 0 and age > τ)
remove this page
if (R == 0 and age ≤ τ)
remember the smallest time

WSClock
(combines Clock and WS Algorithms)

- WSClock
  - Circular list of entries containing
    - R, M, *time of last use*
    - R and *time* are updated on each clock tick
  - Clock "hand" progresses around list
    - If R = 1, reset and update *time*
    - If R = 0, and if *age* > WS_interval, and if clean, then claim it.
    - If R = 0, and if *age* > WS_interval, and if dirty, then schedule a disk write
    - Step "hand" to next entry on list
- Very common in practice

WSClock
(combines Clock and WS Algorithms)



Review of Page Replacement Algorithms

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

Definition

- ***Pinned:*** not subject to being swapped or paged out.
  - *i.e.,* one or more contiguous pages of virtual memory that are stored in specific, identifiable, contiguous page frames in physical memory
  - **i.e. Kernel data** (e.g., PCB's, file objects, semaphores, etc.)

Unix VM

- Physical Memory
  - Core map (pinned) – page frame info
  - Kernel (pinned) – rest of kernel
  - Frames – remainder of memory

- Page replacement
  - Page daemon
    - runs periodically to free up page frames
    - Current BSD system uses ***2-handed clock***
  - Swapper – helps page daemon
    - Look for processes idle 20 sec. or more and swap out longest idle
    - Next, swap out one of 4 largest – one in memory the longest
    - Check for processes to swap in

Linux VM

- Kernel is pinned
- Rest of frames used
  - Processes
  - Page Cache

- Replacement – goal keep a certain number of pages free
  - Daemon (*kswapd*) runs once per second
    - Almost based on the Clock replacement algorithm

Windows

- Processes are assigned *working set minimum* (20-50) and *working set maximum* (45-345)
- Working set minimum is the minimum number of pages the process is guaranteed to have in memory.
- A process may be assigned as many pages up to its working set maximum.
- When the amount of free memory in the system falls below a threshold, *automatic working set trimming* is performed to restore the amount of free memory. (Balance set manager)
- Working set trimming removes pages from processes that have pages in excess of their working set minimum

Review

- If FIFO page replacement is used with four page frames and eight pages, how many page faults will occur with the reference string 0172327103 if the four frames are initially empty? Now repeat this problem for LRU.
- Explain the difference between internal fragmentation and external fragmentation. Which one occurs in paging systems?