# CS 3204
# Operating Systems

Lecture 26

Godmar Back

Virginia Tech

## Announcements

- Project 3 due April 13
- Project 3 help sessions tonight & tomorrow night
  - McB 129 7-9pm
- Project 3 page table design document
  - due Wed March 29
  - only data structures & comments, no code
- If you have bugs left in Project 2, seek help quickly
  - To pass course, must have 95% passing P2 and show reasonable effort on P3 & P4 – can't do that until P2 is done
  - Vijay or I will help you go over your code and point out problems
- Read my 2 posts regarding copy_in issues on forum
- Midterm graded



## Page Replacement

Virginia Tech

## Page Replacement Algorithms

- Goal: want to minimize number of page faults (situations where a page must be brought in from disk.)
  - Also: want to reduce their cost (ideally, evict those pages from their frames that are already on disk – save writeback time)
- Number of algorithms have been developed
  - Global replacement algorithms
    - Treat frames used by all processes equally
  - Local replacement
    - Pool frames according to user or process when considering replacement



## Optimal or MIN Replacement

- To analyze algorithms, consider stream of accesses; each access falls into a given page, e.g.
  2 3 2 1 5 2 4 5 3 2 5 2
- Optimal (also known as MIN, or Belady's algorithm)
  - Replace the page that is accessed the farthest in the future, e.g. that won't be accessed for the longest time
- Problem: don't know what the future holds

| 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 2 | 2 | 2 |
|  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  |  | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |



## FIFO

- Evict oldest page:
  - Problem: completely ignores usage pattern – first pages loaded are often frequently accessed

| 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 5 | 5 | 5 | 5 | 3 | 3 | 3 | 3 |
|  | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 5 | 5 |
|  |  |  | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 2 |

# LRU

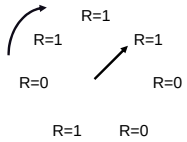
- Evict least-recently-used page
- Great if past = future: become MIN!
- Major problem: would have to keep track of "recency" on every access, either timestamp, or move to front of a list
  - infeasible

| 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| | | | 1 | 1 | 1 | 4 | 4 | 4 | 2 | 2 | 2 |



# Clock

- Also known as NRU (Not Recently Used) or 2nd Chance
- Two ways to look at it:
  - Approximates LRU
  - FIFO, but keep recently used pages
- Use access (or reference bit)
  - R=1 was accessed
  - R=0 was not accessed
- Hand moves & clears R
- Hand stops when it finds R==0






# Clock Example 1

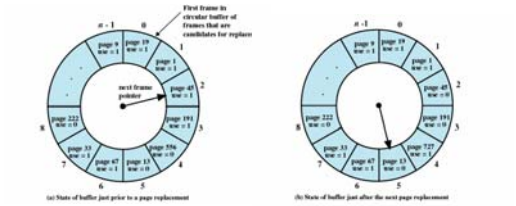


Figure 8.16 Example of Clock Policy Operation



# Clock Example 2

- In this example, assume hand advances only on allocation
  - as you can do in Pintos
- In practice, use clock daemon that periodically scans pages and resets their access bits
  - What if clock daemon scans too fast (all pages look unused)
  - Too slow (all pages look used)

* means R=1 (page was accessed since last scan)

| 2 | 3 | 2 | 1 | 5 | 2 | 4 | 5 | 3 | 2 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2* | 2* | 2* | 2* | 5* | 5* | 5* | 5* | 3* | 3* | 3* | 3* |
| | 3* | 3* | 3* | 3 | 2* | 2* | 2* | 2 | 2* | 2 | 2* |
| | | | 1* | 1 | 1 | 4* | 4* | 4 | 4 | 5* | 5* |



# Variations on Clock Algorithm

- 2-handed Clock
  - If lots of frames, may need to scan many pages until one is found – so introduce second hand
    - Leading hand clears ref bits
    - Trailing hand evicts pages
- Enhanced Clock: exploit modified (or "dirty") bit
  - First find unreferenced & unmodified pages to evict
  - Only if out of those, consider unreferenced & modified pages
  - Clear reference bit as usual



# N-bit Clock Algorithm

- 1-bit says was recently used or wasn't
  - But how recently?
- Idea: associate n-bit counter with page
  - "age" or "act_count"
  - have R-bit as before
- When hand passes page
  - act_count >>= 2 aging
  - act_count |= (R << (n-1)) recent access
- Replace page with lowest act_count