Page Replacement Policy

- When OS needs a frame to use, and all are busy, which page does it evict?
  - Random
    - Pick any page to evict
  - FIFO
    - Evict the page that has been in memory the longest (use a queue to keep track)
  - Optimal (Minimal)
    - Evict the page that will be referenced the farthest into the future
      - Requires knowledge of future
    - Cannot really be implemented
      - Useful for evaluating other policies
  - Least-Recently-Used (LRU)
    - Use the past to predict the future
    - Evict the page that has been unreferenced for the longest period of time

Page Reference Example

- Assumptions: 4 pages, 3 frames
  Page references: ABCABDADBCB

FIFO	A	B	C	A	B	D	A	D	B	C	B
frame 1											
frame 2											
frame 3											

Optimal	A	B	C	A	B	D	A	D	B	C	B
frame 1											
frame 2											
frame 3											

LRU	A	B	C	A	B	D	A	D	B	C	B
frame 1											
frame 2											
frame 3											

Implementing LRU

- A perfect implementation would be something like this:
  - Associate a clock register with every page in physical memory
  - Update the clock value at every access
  - During replacement, scan through all the pages and find the one with the lowest value in its clock register
  - What's wrong with all this?
- Simple approximations:
  - FIFO
  - Not-recently-used (NRU)
    - Use an R (reference) bit, and set it whenever a page is referenced
    - Clear the R bit periodically, such as every clock interrupt
    - Choose any page with a clear R bit to evict

Implementing LRU (cont.)

- Clock / Second Chance Algorithm
  - Use an R (reference) bit as before
  - On a page fault, circle around the “clock” of all pages in the user memory pool
    - Start after the page examined last time
    - If the R bit for the page is set, clear it
    - If the R bit for the page is clear, replace that page and set the bit
  - Questions:
    - Can it loop forever?
    - What does it mean if the “hand” is moving slowly? …if the hand is moving quickly?
- Least Frequently Used (LFU) / N-th Chance Algorithm
  - Don't evict a page until hand has swept by N times
  - Use an R bit and a counter
  - How is N chosen? Large or small?

Frame Allocation

- How many frames does each process get (M frames, N processes)?
  - At least 2 frames (one for instruction, one for memory operand), maybe more…
  - Maximum is number in physical memory
- Allocation algorithms:
  - Equal allocation
    - Each gets M / N frames
  - Proportional allocation
    - Number depends on size and priority
- Which pool of frames is used for replacement?
  - Local replacement
    - Process can only reuse its own frames
  - Global replacement
    - Process can reuse any frame (even if used by another process)

Thrashing

- Consider what happens when memory gets overcommitted:
  - After each process runs, before it gets a chance to run again, all of its pages may get paged out
  - The next time that process runs, the OS will spend a ***lot*** of time page faulting, and bringing the pages back in
    - All the time it's spending on paging is time that it's not getting useful work done
    - With demand paging, we wanted a very large virtual memory that would be as fast as physical memory, but instead we're getting one that's as slow as the disk!
- This wasted activity due to frequent paging is called *thrashing*
  - Analogy — student taking too many courses, with too much work due

Working Sets

- Thrashing occurs when the sum of all processes' requirement is greater than physical memory
  - Solution — use local page frame replacement, don't let processes compete
    - Doesn't help, as an individual process can still thrash
  - Solution — only give a process the number of frames that it "needs"
    - Change number of frames allocated to each process over time
    - If total need is too high, pick a process and suspend it
- *Working set* (Denning, 1968) — the collection of pages that a process is working with, and which must be resident in main memory, to avoid thrashing
  - Always keep working set in memory
  - Other pages can be discarded as necessary

Rules for "The Paging Game"

1. Each player gets several million *things*.
2. Things are kept in *crates* that hold 1024 things each. Things in the same crate are called crate-mates.
3. Crates are stored either in the *workshop* or the *warehouse*. The workshop is almost always too small to hold all the crates.
4. There is only one workshop but there may be several warehouses. Everybody shares them.
5. Each thing has its own *thing number*.
6. What you do with a thing is to *zark* it. Everybody takes turns zarking.
7. You can only zark your things, not anybody else's.
8. Things can only be zarked when they are in the workshop.
9. Only the *Thing King* knows whether a thing is in the workshop or in a warehouse.
10. The longer a thing goes without being zarked, the *grubbier* it is said to become.

Rules for “The Paging Game” (cont.)

11. The way you get things is to ask the Thing King.
12. The way you zark a thing is to give its thing number. If you give the number of a thing that happens to be in a workshop it gets zarked right away. If it is in a warehouse, the Thing King packs the crate containing your thing back into the workshop. If there is no room in the workshop, he first finds the grubbiest crate in the workshop, whether it be yours or somebody else’s, and packs it off with all its crate-mates to a warehouse. In its place he puts the crate containing your thing. Your thing then gets zarked and you never know that it wasn’t in the workshop all along.
13. Each player’s stock of things have the same numbers as everybody else’s. The Thing King always knows who owns what thing and whose turn it is, so you can’t ever accidentally zark somebody else’s thing even if it has the same thing number as one of yours.

Notes on “The Paging Game”

1. Traditionally, the Thing King sits at a large, segmented table and is attended to by pages (the so-called “table pages”) whose job it is to help the king remember where all the things are and who they belong to.
2. One consequence of Rule 13 is that everybody’s thing numbers will be similar from game to game, regardless of the number of players.
3. The Thing King has a few things of his own, some of which move back and forth between workshop and warehouse just like anybody else’s, but some of which are just too heavy to move out of the workshop.
4. With the given set of rules, oft-zarked things tend to get kept mostly in the workshop while little-zarked things stay mostly in a warehouse. This is efficient stock control.

Long Live the Thing King!