11/7 Eviction

Limited Memory

→ Physical memory is full, what do we do?

① Out of memory killer
    → disruptive in desktop setting
    → more common in mobile OSes

② Reuse physical frames
    → storage devices (disk, SSD, larger in size)
    → eviction: select a frame to evict, write its content to storage,
give the frame to another page

☆ Do we always need to write the frame out to disk?
→ code a clean data page (already in Elf file on disk)
→ memory mapped files

Why? B/c it might be accessed later!
iOS developers trying to figure out memory budget to avoid their apps from being killed.

---

device: (crash amount/total amount/percentage of total)
- iPhone X: 1392/2785/50% (iOS 11.2.1)
- iPhone XS: 2040/3754/54% (iOS 12.1)
- iPhone XS Max: 2039/3735/55% (iOS 12.1)
- iPhone XR: 1792/2813/63% (iOS 12.1)
- iPhone 11: 2068/3844/54% (iOS 13.1.3)
- iPhone 11 Pro Max: 2067/3740/55% (iOS 13.2.3)
- iPhone 12 Pro: 3054/5703/54% (iOS 16.1)

---

3 iPhone 5 crashes at ±645 MB. – asp_net Dec 15, 2013 at 21:03

iPhone 5S crashes at ±646 MB pretty reliably here. – eAi Oct 3, 2014 at 13:30

iPhone 4S crashes at ±286MB (286MB/512MB/56%). – Xaree Lee May 29, 2015 at 21:50

iPhone 4S doesn’t crash until it reaches ±363 MB for me. (iOS 5.1.1) – Soeholm Jun 3, 2015 at 16:53

2 Awesome that this list has been created and maintained. In my experience, I’ve had to keep memory much lower to be safe, maybe 20% of what’s shown here. Device-to-device differences are also highly variable. – user1021430 Aug 10, 2015 at 19:24

1 Just ran this on a 12.9 iPad Pro. Memory warning at 2451MB, crash at 3064MB, total 3981MB. – Lock Jul 15, 2016 at 13:22

iPhone 6s+: 1392MB/2048MB/ 68% (iOS 10.2.1); iPhone 7+: 2040MB/3072MB/66% (iOS 10.2.1) – Slyv Feb 13, 2017 at 15:39
Eviction Mechanisms

→ Where do we write the evicted page to?
  • Swap partition (or file)
    → section of sectors or blocks
  • Swap allocation (track which sectors/blocks are free, bitmap)
  • Update bookkeeping structures so we can serve page fault for the evicted page

→ How to reflect the eviction
  • remove the old mapping (TLB shutdown)
  • zero out the frame (prevent info leaking)
  • install the new mapping

Can multiple pages be mapped to a single frame?
  → shared memory
  → cow fork

It needs to track refcount "core_map_entry" tracks info about each frame
Eviction Policies

→ What frames to choose from?

→ Global page replacement (eviction) policy
  → Can evict any frame in physical memory
  → A process can affect other processes' performance by evicting their pages out

→ Local page replacement policy
  → Only evict frames occupied by the current process, doesn't affect other processes
  → Need to decide on how many frames each process can have
Policies

*Goal:* select page to evict that will result in less page faults in the future

→ **LRU**: least recently used, if not used recently, less likely to be used in the future, so good candidate for eviction

→ How to implement this?
  → Sw: queue, easy to do
  → Hw: timestamp ??

→ Much cheaper to approximate LRU in Hw instead

→ What access patterns might be bad for LRU?
  → N frames, N+1 pages (iterate through every page a couple of times)

<table>
<thead>
<tr>
<th>Frame</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>
Clock (Approximate LRU)

- clock hand: starting frame for eviction
- check frame’s page’s page table entry
  - pte stores permission & access info (bit 5 = access bit)
- access bit = 0: select frame to evict, advance clock hand
- access bit = 1: clear bit and move on (keep looking)

Clock only considers timing info, but recall that some frames might not need to get written back to disk, picking those frames could make eviction much faster.