11/6/23

Page Fault: valid vs. invalid fault
- ↳ should handle this case by allocating a frame
  but what if there are no more frames?

Running out of frames

-> "make room for it"
- ↳ pick a frame, with its data to storage, reallocate the frame

large but slow

(page eviction)

-> "kill a process"
- ↳ common in mobile OSes
- ↳ mobile apps built to tolerate crashes

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)

iOS developers trying to figure out memory budget for their apps to avoid being killed

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

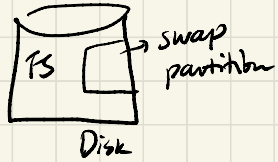
iPhone 4S doesn't crash until it reaches ±363 MB for me. (iOS 5.1.1) – Soeholm Jun 3, 2015 at 16:53

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

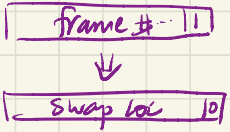
How to evict?



-> assume we already found a page to evict
-> write the frame data to storage device
  -> swap partition (a section on disk that stores data from evicted pages)
  -> swap allocation (typically a bitmap is used for tracking block usage within swap)

Eviction Steps (after finding the page to evict)

1. remove the page mapping from the page table (clear out the present bit of the PTE), flush TLB
2. find available blocks in swap, update swap bitmap, write the page out
3. update per page metadata to track its swap location



What if multiple pages are mapped to the same frame?
  may happen as a result of COW, shared memory
  need to clear the page table entry for all pages mapped to the same frame
  how do we find these pages? frame metadata (xk kernel/kalloc.c: core_map_entry)

After writing out the evicted page, we can now re-allocate the frame
-> need to erase the old frame's content before mapping the faulting page to it
-> zero out the old frame, or load in content of the faulting page if applicable
(e.g. page was evicted before)
-> done handling the page fault, returns from exception, retries the memory access

When to evict?
-> when you have to, no more available frames
-> set a threshold of free frames and start evicting pages when falling below the threshold
(may do this in the background, avoid having to pay the cost of eviction on page fault)

What page to evict?

Who are the eviction candidates?
-> global policies: any mapped pages can be evicted (more options)
-> local policies: can only evict your own pages (pages within the faulting process)
need to set frame limits per process

→ Linux

→ Windows combo of global & local

Cost of page eviction
-> write data to disk
-> if the page has not been written to, don't have to write to disk
-> if the page is a code page, don't need to write to disk (already on disk)