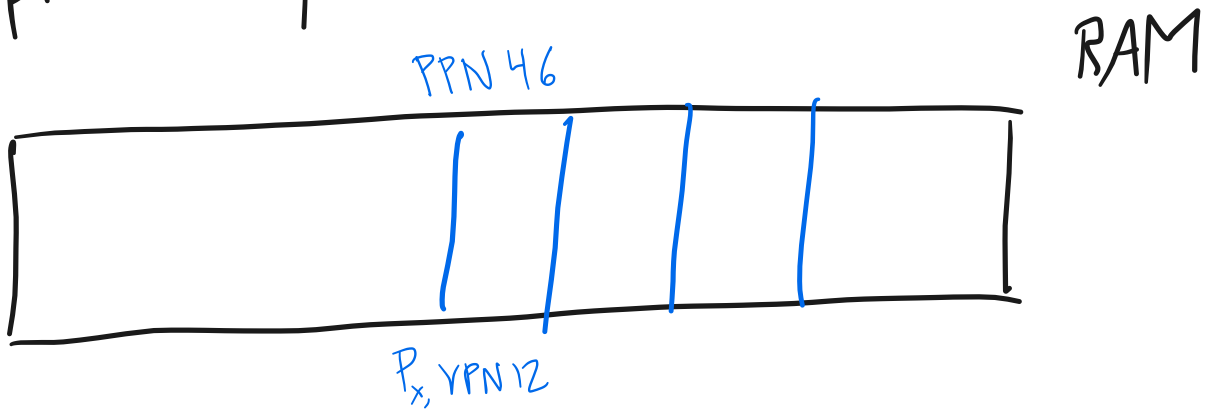
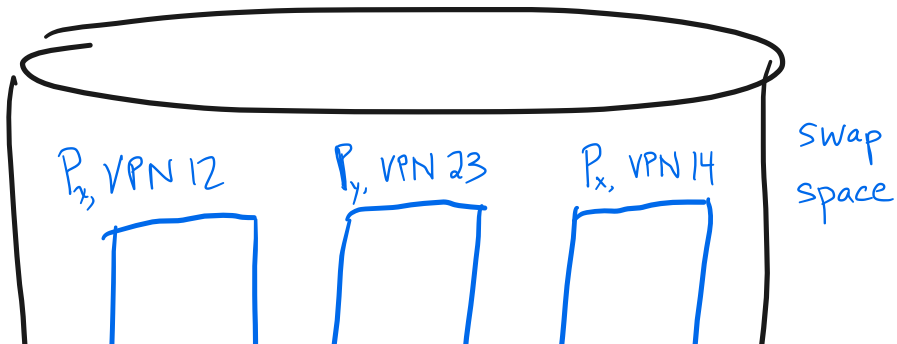
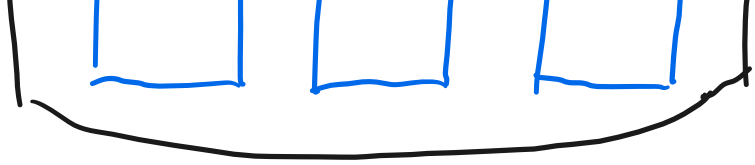
- [x] 1. Last time
- [x] 2. Page replacement policies
- [x] 3. Thrashing
- [ ] 4. Where does the OS live?
- [ ] 5. mmap()

---

2. Page replacement policies







- FIFO: eject oldest
- MIN (OPT): eject entry that won't be referenced for the longest time

input:
reference string
cache size

output:
number of evictions, or more generally misses

FIFO

	A	B	C	A	B	D	A	D	B	C	B
phys_slot											
S1	A			h		D		h		C	
S2		B			h		A				
S3			C						B		h

4 hits, 7 misses

OPTIMAL

	A	B	C	A	B	D	A	D	B	C	B
phys_slot											
S1	A	h		h					h		h
S2		B		h							
S3			C			D	h				

6 hits, 5 misses

LRU

	A	B	C	A	B	D	A	D	B	C	B
phys_slot											
S1	A	h		h					h		h
S2		B		h							
S3			C			D	h				

6 hits, 5 misses

LRU

	A	B	C	D	A	B	C	D	A	B	C	D

S1 A ~~D~~ C D

S2 B A D C

S3 C D B A A

12 misses

---

back to FIFO

A B C D A B E A B C D E

S1 A D E h

S2 B A h C

S3 C B h D

3 hits, 9 misses

A B C D A B E A B C D E

S1 A h E D

S2 B h A E

S3 C B

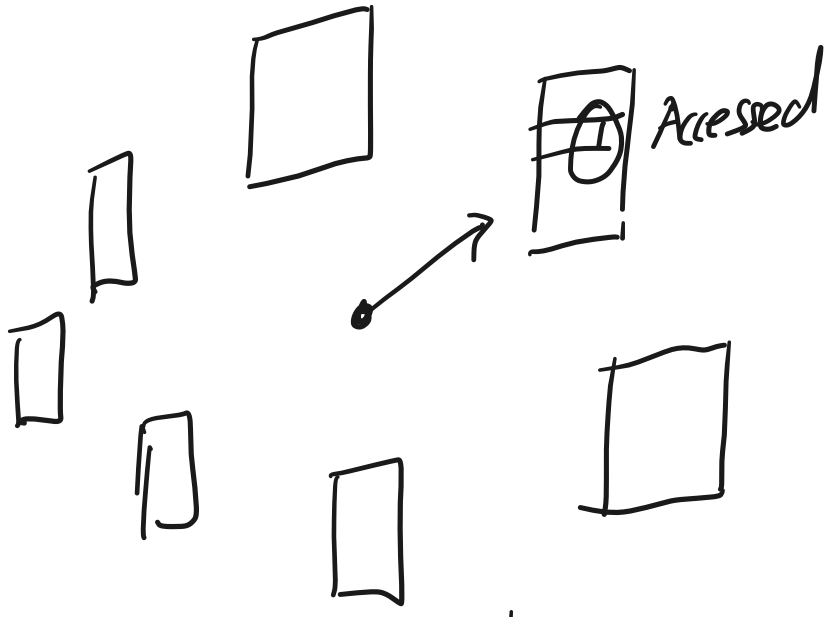
S4 D C

2 hits, 10 misses

- OPT minimizes misses/swaps/evictions
  - but can't be implemented in general.

- LRU: approximates OPT (assuming what?)

- approximate LRU with CLOCK



H/W sets Accessed + Dirty bits

OS consumes these bits and clears them.

- Generalization of CLOCK: N^{th} Chance (see notes).

3. Thrashing

ex: program touches 50 pages, equiprobably
but only 40 phys. frames (or slots)

Thrashing: processes demand more memory for active use than the system has.

3 reasons:
(a) process has no temporal locality, or
(b) " " temporal locality but not enough memory; or
(c) individually all processes fit, but there's not enough memory.

---

stats

median: 47/100

μ : 47.9

σ : 26.7

high $\approx$ 100

$A \approx 75$

low ≈

Virtual Address Space of a Linux Process