Virtual Memory and Paging (2) (Practice)

ICS332

Operating Systems

Henri Casanova (henric@hawaii.edu)

(q1) Reconstructing Page Table

- Say a system uses a single-level page table, with 1000-byte pages (pages and frames are numbered starting at 0), and we freeze time at a point there the running program has had no page evicted back to disk
- We know the following facts:
  - Accessing logical address 42312 would page fault
  - Logical address 1451 translates to physical address 17451
  - Logical address 5601 translates to physical address 12601
  - No byte between physical addresses 11500 and 13050 has ever been written to
  - Logical address 6501 has been written to at least once
- Fill in all the we know in the page table below:

(q1) Answer

- Say a system uses a single-level page table, with 1000-byte pages (pages and frames are numbered starting at 0), and we freeze time at a point there the running program has had no page evicted back to disk
- We know the following facts:
  - Accessing logical address 42312 would page fault
  - Logical address 1451 translates to physical address 17451
  - Logical address 5601 translates to physical address 12601
  - No byte between physical addresses 11500 and 13050 has ever been written to
  - Logical address 6501 has been written to at least once
- Fill in all the we know in the page table below:

Page number	Frame Number	Valid bit	Dirty bit
42		0	
1	17	1	
5	12	1	0
6		1	1

(q2) Page Eviction

- Consider the following page reference sequence 0,5,0,1,7,0,2,0 that occurs on a system with 3 physical frames only allocated to the program that issues these page references
- Show the RAM content and page faults at each step, assuming no page is in RAM initially, and using the **FIFO** algorithm

page ref	0	5	0	1	7	0	2	0
frame #0								
frame #1								
frame #2								
frame #3								
page fault								

(q2) Answer

- Consider the following page reference sequence 0,5,0,1,7,0,2,0 that occurs on a system with 3 physical frames only allocated to the program that issues these page references
- Show the RAM content and page faults at each step, assuming no page is in RAM initially, and using the **FIFO** algorithm

page ref	0	5	0	1	7	0	2	0
frame #0	0	0	0	0	7	7	7	7
frame #1		5	5	5	5	0	0	0
frame #2				1	1	1	2	2
frame #3								
page fault	✓	✓		✓	✓	✓	✓	

(q3) Page Eviction

- Consider the following page reference sequence 0,5,0,1,7,0,2,0 that occurs on a system with 3 physical frames only allocated to the program that issues these page references
- Show the RAM content and page faults at each step, assuming no page is in RAM initially, and using the **LRU** algorithm

page ref	0	5	0	1	7	0	2	0
frame #0								
frame #1								
frame #2								
frame #3								
page fault								

(q3) Answer

- Consider the following page reference sequence 0,5,0,1,7,0,2,0 that occurs on a system with 3 physical frames only allocated to the program that issues these page references
- Show the RAM content and page faults at each step, assuming no page is in RAM initially, and using the **LRU** algorithm

page ref	0	5	0	1	7	0	2	0
frame #0	0	0	0	0	0	0	0	0
frame #1		5	5	5	7	7	7	7
frame #2				1	1	1	2	2
frame #3								
page fault	✓	✓		✓	✓		✓	

(q4) Clock Algorithm

- Say we have a system with 8 frames, and that right now, the reference bits are as follows: 0 1 0 0 1 1 0 0
- Say that the next references by the running program reference frames #0, #5, and #7
- After that, the process allocates memory and the OS doesn't give it a new frame, so that a frame needs to be evicted
- Which frame is evicted?

(q4) Answer

- Say we have a system with 8 frames, and that right now, the reference bits are as follows: 0 1 0 0 1 1 0 0
- Say that the next references by the running program reference frames #0, #5, and #7
- New reference bits: 1 1 0 0 1 1 0 1
- After that, the process allocates memory and the OS doesn't give it a new frame, so that a frame needs to be evicted
- Which frame is evicted? Frame #2
- New reference bits: 0 0 1 0 1 1 0 1