COMP 530 Introduction to Operating Systems

Fall 2017  
Kevin Jeffay

Worksheet 16, October 30

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Your Name: |  | You worked with: |  | +1/blank/-1: |  |
|  | Aaron Zhang |  | John Espenhahn |  | +1 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. Consider a processor that has 3 reference bits for each entry in the page table rather than the single reference bit we assumed in class. Described how one could come up with a better implementation of LRU page replacement with 3 reference bits per page table entry. Explain why you would expect that your implementation would be better (and explain what you mean by “better”)If a processor has 3 reference bits for an entry on the page table, then we could technically increment the reference counter of a page for up to 7 times if we access the same page 7 times. Considering that the LRU system with only 1 bit, a page is only least recently used with respect to a single sweep of the clock hand. Now, 3 reference bits allow a better approximation with LRU because a page that was accessed for 2 times can now be compared with a page that was only last accessed once and hence the page which was less frequently accessed will be replaced. (As compared to one reference bit, where a twice accessed page has same bit number as a once accessed page)  
  
We could modify the implementation by changing the 'reset used bit' line in the code to 'decrement reference bit by 1'.  
  
This implementation is better because it allows us to compare the access frequency of pages more accurately (e.g a page that was repeatedly accessed 2 times vs a page that was last accessed once) and thus this leads to better page replacement.

2. Consider a demand paged virtual memory operating system that allocates 5 memory frames to every process and replaces pages according to the clock page replacement algorithm. Consider the page reference string:

*a, b, c, d, e, f, g, c, e, a, f, b, c, f, h, d*

Assuming the memory allocated to the process is initially empty, determine the number of page faults that occur for this reference string. Show how you derived this number. In particular, make it clear (1) which page references (if any) cause page faults, and (2) which pages are replaced at which times.

When switching between processes, the number of page faults will be reduced. When forking, the parent and child page tables have the same mappings. Instruction memory doesn’t change, so no need to make a literal copy of that code.