Ayça Avcı
53961

13/05/2020

COMP 304
Assignment #3

Problem 1:

In contiguous allocation, best-fit and first-fit methods suffer from external fragmentation. It does not allow code sharing among processes since a process's virtual memory segment is not broken into non-contiguous fine grained segments. Segmentation suffers from external fragmentation and allows code sharing among processes. In segmentation, external fragmentation can occur when segment of dead processes are replaced by segment of new processes since segment of a process is laid out contiguously in physical memory. In segmentation, two different processes can share a code segment but have distinct data segments. Paging does not suffer from external fragmentation and it allows code sharing among processes.

Problem 2:

$8\,KB \times 1024 = 2^{13} \rightarrow 13$ bit = page offset

($8\,KB \sim 2^3$, $1024 \sim 2^{10}$)

$32 - 13 = 19$ bit = page number

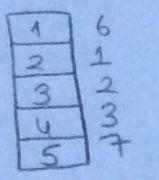
$2^{19} \times 4$ bytes $= 2\,MB$ = total size of page table

(2^{19}: pages in address space; 4 bytes: each entry)

Problem 3:

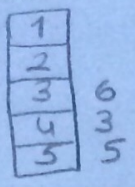
a)

FIFO



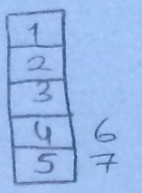
10 page faults

LRU



8 page faults

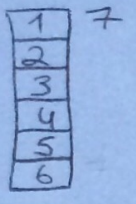
Optimal



7 page faults

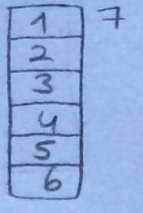
b)

FIFO



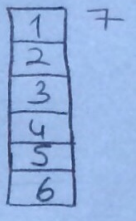
7 page faults

LRU



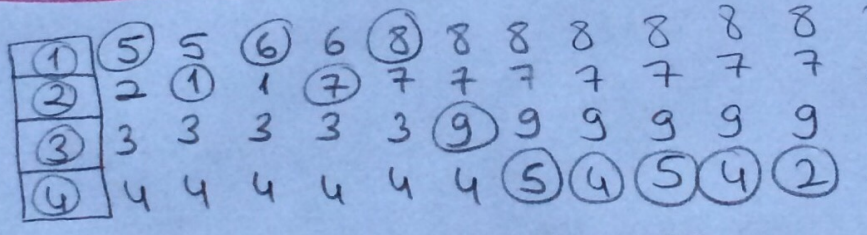
7 page faults

Optimal



7 page faults

Problem 4:



15 page faults

1: 1 0 1 0 2: 1 0 1 1 3: 1 1 0 4: 1 2 0 1 0 5: 1 0 1 0 0 6: 1 0 7: 1 1 3 8: 1 1 3 9: 1 2

(LFU algorithm applied, if counters are equal FIFO used.)

It is not a good algorithm. It suffers from the situation when new items just entered the cache, they are subject to being removed very soon again, because they start with low counter, even though they might be used very frequently after that. Hybrid solutions can be used to utilize LFU concepts.

Problem 5:

a) Let F1 be the old file and F2 be the new file. A user wishing to access F1 through an existing link will actually access F2. Also, access protection for F1 is used instead of the one associated with F2.

b) Problem can be solved by insuring that all the links to a deleted file are deleted also. It can be achieved by maintaining a list of all links to a file, removing each of them when the file is deleted.

Problem 6:

a) P1: {1, 2, 3, 4, 7} → size = 5
P2: {3, 4, 5, 6} → size = 4
P3: {1, 2, 7, 8, 9} → size = 5

D = Sum of WSSi
= total demand of frames
= 14
m = total number of frames
= 10
D > m ⇒ trashing
↳ 14 > 10 ↗

b) System suffers from trashing. We have total 14 pages to insert to 10 frames. 10 frames are not enough to avoid trashing in that case since processes are independent and cannot share data.

c) P1: {1, 2, 3, 4, 5, 7}
P2: {1, 3, 4}
P3: {2, 3, 7, 8, 9}

We have 6+3+5 = 14 pages. So, we need 14 frames to prevent trashing.

Problem 7:

For program 1, each iteration of i will generate a page fault for every value of j. In the first access, it will give a page fault for first row, but when it access the same row again, page will not be there due to LRU algorithm. It will lead to a page fault again. Therefore, total page fault is $1024 \times 1024 = 1048576$.

For program 2, it will generate a page fault in its first access. When it access again, page will be alread there. So, till i increases by 1, page fault will occur only once because of the first access for some $i=k$. (k is $0 \leq k \leq 1023$) Therefore, total page fault is 1024.

Problem 8:

a) Because every address a process uses is translated by its page table, so it cannot express address in another process.

b) 1. Set up entries for physical pages used by other processes in its page table.
2. Provide a system call for r/w from other processes.