Comp304-Assignment3

İrem Demir
69563

Problem-1:

External fragmentation:

External fragmentation happens when enough memory in total can be found but that memory is not contiguous. Only paging does not suffer from this problem.

Contiguous Allocation: Processes get space that they require to execute. When processes get and release memory, there might be little free pieces in the memory. All of the best, first and worst fit algorithms for dynamic storage-allocation problem suffers from this. Even though when a process allocated with worst fit produces largest leftover, this is still an issue. External fragmentation can be reduced by compaction which shuffles memory contents to get all free memory in one place. However, it only reduces this and is possible if relocation is made dynamically.

Segmentation: Processes get space that they require to execute as segments. This method can suffer from external fragmentation. As in contagious allocation, process can divide memory into small pieces.

Paging: Pages are fixed size partitions of virtual memory in order. Also for each page there is a fixed size partition in physical memory. Since memory divided into "pages" of fixed-sizes no external fragmentation can be occur in paging.

Code Sharing:

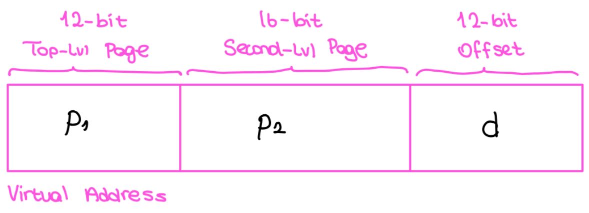
Contiguous Allocation: This does not allow code sharing.

Segmentation: Processes can use same segments. Thus code sharing is possible via sharing segments. Since a segment is a logical unit such as a main program, function, objects it is useful and easy to reason for code sharing.

Paging: Processes can use same pages. Thus code sharing is possible via sharing pages. Moreover, Copy-on-Write allows parent and child processes to share same pages is memory when both did not change the shared page.

Problem-2:

The memory address will have the following structure:



- Page size will be determined by the offset since each "starting point" will be coded by that. Thus, there will be 2^{12} such location and each will have size 1 byte.

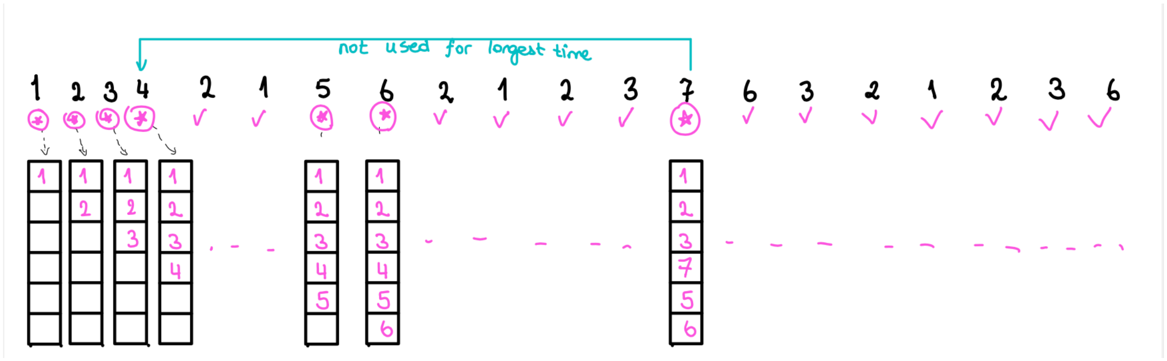
$$\rightarrow \text{Page Size}: \quad 2^{12}\,\text{byte} = 4\,\text{KB}$$

- Total page number is determined by page table fields. Top level page field will act as a page directory and each one will point to a page table in which we will use second level page table field to select page. Thus, top one will have 2^{12} entry that points to a page table that has 2^{16} page reference. Then

$$\rightarrow \#\text{Page}: \quad 2^{12} \times 2^{16} = 2^{28} \,//$$

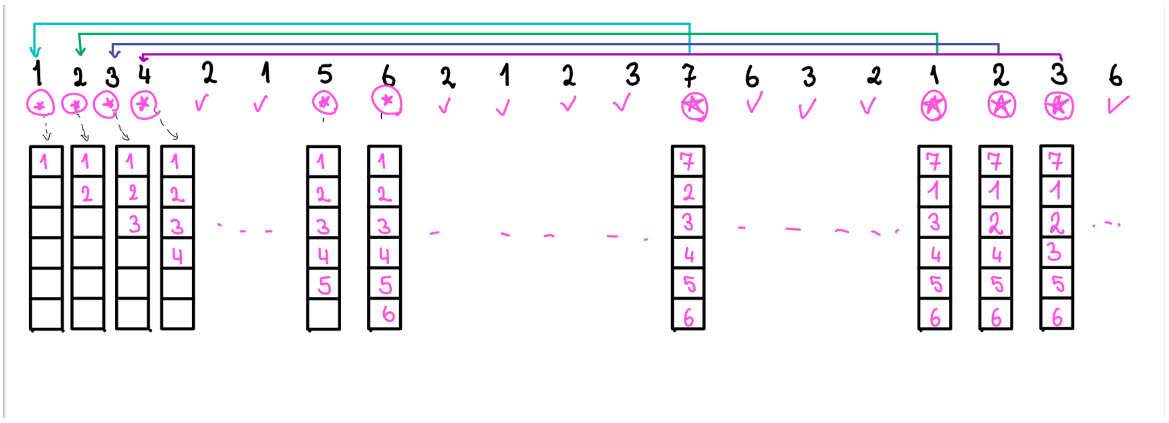
Problem-3:

LRU:



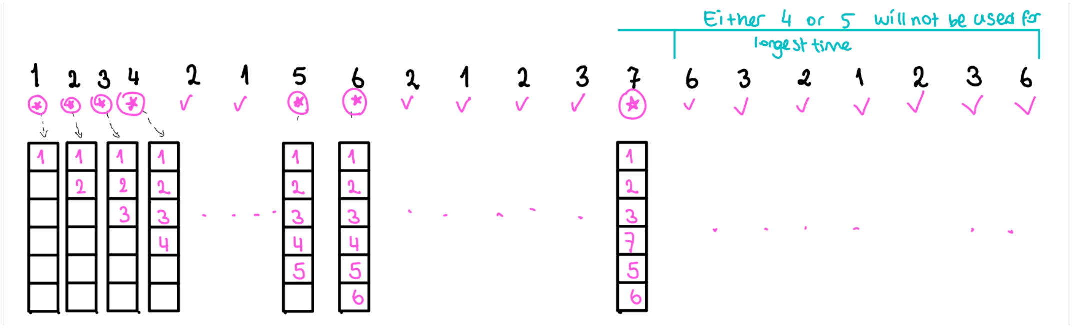
Total #Page Faults: 7

FIFO:



Total #Page Faults: 10

OPTIMAL:



Total #Page Faults: 7

Problem-4:

A) file1.txt has inode value: 35061447

file2.txt has inode value: 35061447 as well.

They both have the same content.

B) After updating content of file2.txt, the content of file1.txt is changed as well. So, they both have same content in the end. Their inode values are same as well.

C) After removing file1.txt, file2.txt still exists.

D) The system call removed file2.txt is unlinkat()

E) file3.txt has inode value: 35061460

file4.txt has inode value: 35063743. So their inode values are unique.

F) After updating the content of file4.txt, the content of file3.txt is changed as well. So, they both have same content in the end.

G) The content of file4.txt is emptied. After adding new elements, file3.txt is created with the content of file4.txt again.

H) Hard links refers to same inode so that they refer to same data. However, soft link just points to original file. When there is a change in the content both in soft and hard links all files are updated since for soft links the pointed file is changed and for hard links the data changed. However, the difference comes to surface in removing files. In hard links, removing original file does not affect the linked file. Since it still refers the inode that stores the data and there is no change in there. On contrary, when the original file deleted in soft link, linked content is also gone because now it points to an emptied/deleted file. Moreover, the original file is created again when we change the content of linked file since it changes the contents of original file and now there exists a data in there and it created again. One usage of hard links might be file backup in the same file system. If something happens to "original file" that cause its deletion we can still achieve its latest saved version using hard linked file. Soft links are like shortcuts. It can be used in such cases. Moreover, they do not depend on file system so they can be used in different ones as opposed to hard links. Another usage is they can be used to access files with new aliases.