

Design Document

I. What happens during a page-fault?

- During the page fault, I first check if it is a user virtual address using the function in `vaddr.h` along with checking that if the virtual address is not present. If the condition is not satisfied, It will generate page fault exception and kill the process. If the condition is satisfied, it will search into supplemental page table for the fault address, if there is supplemental page table entry associated with that fault address, then it will load the page from the supplemental page table structure. If entry not present, it will check if the fault address is asking for a stack growth, then grow the stack if appropriate.
- Struct `spage_entry`
 - o Void `*user_va`: user virtual address the entry is holding
 - o Int `spage_type`: the page type dictates in which functionality to run
 - o Bool `is_pinned`: pin the page to not evict
 - o Bool `is_loaded`: the page is loaded or not
 - o Bool `writable`: if the page is writable or not
 - o Struct `file *file`: lazy file loading
 - o Size_t `offset`: lazy file loading
 - o Size_t `read_bytes`: lazy file loading, precalculated from `load_segment`
 - o Size_t `zero_bytes`: lazy file loading, pre-calculated from `load_segment`
 - o Siset `swap_index`: index to be used to identify swap
 - o Struct `hash_elem elem`: hashable

II. How did you implement eviction & pinning?

- `frame_evict` will be called when it cannot retrieve page using `palloc`. Then the eviction will be repeatedly check for the swappable frame in the frame table. That is, if there is frame table entry that page was not pinned, it will check if the page was accessed, and if it is accessed before, it will set it so that next time coming back, it can be accessed. Then it will check page's dirtiness if not accessed before. If the page is dirty, it will swap out. Then return the freed page.
- Struct `Frame_table_entry`
 - o Void `*frame`: the pointer to frame
 - o Struct `spge_entry`: supplemental page table entry that frame is associated with
 - o Struct `thread *t`: thread that frame was being used
 - o Struct `list_elem`: listable
- Swap is done by frame table entry and `page_dir`.
- `Swap.h` contains struct `bitmap swap_bitmap` to keep track of free swap slots
- `Swap.h` contains struct `blok *swap` as suggested in the project slides

III. How do you implement memory mapped files?

- a. System call handler code
- b. Data structures to keep track of them

- IV. How did you implement page sharing
- I did not implement page sharing functionality