---- TEAM ----

**>> Team name.**

42

**>> Fill in the names, email addresses and contributions of your team members.**

Ina Ryu<ryu\_ina@kaist.ac.kr> (50)

Yeonsoo Kim <kysoo55@kaist.ac.kr> (50)

contribution1 + contribution2 = 100

**>> Specify how many tokens your team will use.**

PJ3-1 : token 1

                                          PAGE TABLE MANAGEMENT

                                          ===================

---- DATA STRUCTURES ----

**A1: Copy here the declaration of each new or changed `struct' or `struct' member, global or static variable, `typedef', or enumeration.  Identify the purpose of each in 25 words or less.**

<supplemental page table (list) (global) >

struct supplemental page table entry  {

struct list\_elem elem; /\* List element \*/

void \*vaddr; /\* virtual address \*/

void \*paddr; /\* physical address \*/

tid\_t pid; /\* process's pid that allocate this frame \*/

bool is\_exec; /\* true : executable file \*/

};

<frame table (list) (global) >  
struct frame table entry

{

struct list\_elem elem; /\* List element \*/

void \*paddr; /\* physical address \*/

void \*vaddr; /\* points virtual address page \*/

tid\_t pid; /\* process's pid that allocate this frame \*/

};

<swap table (hash) (global) >

struct swap table entry {

struct hash\_elem hash\_elem; /\* Hash table element \*/

disk\_sector\_t disksector; /\* disk sector num \*/

void \*vaddr; /\* swap out 시킨 vaddr

  if not swapped : vaddr = NULL \*/

tid\_t pid; /\* process's pid that allocate this frame \*/

};

---- ALGORITHMS ----

**A2: In a few paragraphs, describe your code for locating the frame, if any, that contains the data of a given page.**

pagdir\_set\_page() 함수가 page에 frame을 배정하면,  s\_pte\_insert()라는 함수로 supplemental page table에 그 연결을 추가한다(vaddr, paddr, pid). 그리고 그 다음 frame table에도 연결을 추가한다(vaddr, paddr, pid).

추후에 pagedir\_clear\_page 할 때도, supplemental page table에는 paddr 을 NULL로 해주고, frame table 에서는 entry 를 지워준다. 왜냐하면 supplemental page table 에서는 mapping이 유지되어야 나중에 page fault 되었을 때 이게 swapout 된건 지 알 수 있기 때문이다.

**A3: How does your code coordinate accessed and dirty bits between kernel and user virtual addresses that alias a single frame, or alternatively how do you avoid the issue?**

supplemental page table을 global로 만들었고, 각 연결에 process id를 추가했다. 따라서 supplemental page table을 참고해서 다른 process의 page table을 참조할 수 있다.

---- SYNCHRONIZATION ----

**A4: When two user processes both need a new frame at the same time, how are races avoided?**

모든 frame 에 대한 접근은 frame\_table에 의해 정해지는데, frame\_lock이라는 lock을 설정하여, frame table에 접근할때마다 lock\_acquire을 하고, 완료하면 lock\_release를 해 avoid한다.

---- RATIONALE ----

**A5: Why did you choose the data structure(s) that you did for representing virtual-to-physical mappings**?

모든 table에는 virtual page주소와 physical frame주소 둘다 저장하여, 모든 곳에서 참고하기 쉽게 했다.

그리고 supplemental page table은 list 형태로 만들어줬는데, evict policy로 FIFO를 사용했기 떄문에 list형태가 좋았다.

                              PAGING TO AND FROM DISK

                              ====================

---- DATA STRUCTURES ----

**B1: Copy here the declaration of each new or changed `struct' or `struct' member, global or static variable, `typedef', or enumeration.  Identify the purpose of each in 25 words or less.**

<swap table (hash) (global) >

struct swap table entry {

struct hash\_elem hash\_elem; /\* Hash table element \*/

disk\_sector\_t disksector; /\* disk sector num \*/

void \*vaddr; /\* swap out 시킨 vaddr

  if not swapped : vaddr = NULL \*/

tid\_t pid; /\* process's pid that allocate this frame \*/

};

---- ALGORITHMS ----

**B2: When a frame is required but none is free, some frame must be evicted.  Describe your code for choosing a frame to evict.**

fifo 알고리즘 사용.

supplemental page table에 추가할 때에는 무조건 list\_push\_front로 넣어주고, evict 할 frame을 고를 때에는 list\_pop\_back()을 사용하여 맨 뒤에있는 것을 골라준다.

**B3: When a process P obtains a frame that was previously used by a process Q, how do you adjust the page table (and any other data structures) to reflect the frame Q no longer has?**

supplemental page table은 global 해서 각 entry에 pid도 저장한다. 그리고 한 page를 evict 하면 그 엔트리의 pid를 통해 pagedirectory를 찾아서 pagedir\_clear\_page를 해준다. 그래서 P가 Q가 쓰는 page를 가져가면 Q의 pagedir도 변경시켜줘서 적용할 수 있다.

**B4: Explain your heuristic for deciding whether a page fault for an invalid virtual address should cause the stack to be extended into the page that faulted.**

fault address와 page fault 가 발생한 fault addes 비교했다.

fault\_addr > f->esp - 32이면 stack growth이고, 그것이 아니라면 stack growth가 아닌 경우이다.

---- SYNCHRONIZATION ----

**B5: Explain the basics of your VM synchronization design.  In particular, explain how it prevents deadlock.  (Refer to the textbook for an explanation of the necessary conditions for deadlock.)**

frame\_lock : frame table에 하나씩만 접근할 수 있도록

s\_pt\_lock : supplemental page table 에 하나씩만 접근할 수 있도록

swap\_lock : swap table (disk 접근 : swap in, swap out) 에 하나씩만 접근할 수 있도록

evict\_lock : evict 과정을 한번에 하나씩만 할 수 있도록 + evict 하는 도중에 process가 free가 되지 않게함.

**B6: A page fault in process P can cause another process Q's frame to be evicted.  How do you ensure that Q cannot access or modify the page during the eviction process?  How do you avoid a race between P evicting Q's frame and Q faulting the page back in?**

evict\_lock을 사용하여 한 process가 evict하는 동안 다른 process의 접근을 막았다.

**B7: Suppose a page fault in process P causes a page to be read from the file system or swap.  How do you ensure that a second process Q cannot interfere by e.g. attempting to evict the frame while it is still being read in?**

swap\_in 또한 evict\_lock으로 둘러쌓여 있어서, evict\_lock 때문에 swap\_in과 eviction이 동시에 일어날 수 없다.

**B8: Explain how you handle access to paged-out pages that occur during system calls.  Do you use page faults to bring in pages (as in user programs), or do you have a mechanism for "locking" frames into physical memory, or do you use some other design?  How do you gracefully handle attempted accesses to invalid virtual addresses?**

page를 배정해주어야 하는 경우엔 page\_fault로 넘겨 주었고, 배정해줄 필요 없이 process exit 해야하는 경우에는 바로 sys\_exit(-1)을 해주었다.

---- RATIONALE ----

**B9: A single lock for the whole VM system would make synchronization easy, but limit parallelism.  On the other hand, using many locks complicates synchronization and raises the possibility for deadlock but allows for high parallelism.  Explain where your design falls along this continuum and why you chose to design it this way.**

우리는 많은 lock을 만들어주어 한 과정당 하나의 process가 할 수 있도록 했다. 그래서 각 과정이 같은  data structure에 접근하는게 아니면 parallel하게 돌아간다. 대신 deadlock에 걸리지 않도록, 순서를 조정하였다.

                                          MEMORY MAPPED FILES

                                          ===================

---- DATA STRUCTURES ----

**C1: Copy here the declaration of each new or changed `struct' or ‘struct' member, global or static variable, `typedef', or enumeration.  Identify the purpose of each in 25 words or less.**

struct process

{

struct list load\_file\_table; /\* Manage exec file of this process \*/

struct list mapping\_list; /\* Manage mmap memory \*/

void \* stack\_end; /\* Point end of stack \*/

void \* stack\_start; /\* Point end of stack \*/

};

<mmap-table>

struct mapping {

struct list\_elem elem; /\* List element \*/

void \* start; /\* start (virtual address) linear pages\*/

uint32\_t size; /\* total page size \*/

int id; /\* mapping ID 1,2,.... \*/

struct file \*file; /\* mapped file \*/

int fd; /\* mapped file descriptor \*/

struct list file\_table; /\* File table for each entries \*/

};

<file table>

struct fte {

struct list\_elem elem; /\* List element of file\_table\*/

void \* vaddr; /\* virtual page address \*/

off\_t ofs; /\* Offset from file \*/

uint32\_t size; /\* read size \*/

bool writable; /\* writable \*/

};

<s-pagetable>

struct s\_pte {

int mmap\_id; /\* 0 : lazy loading, 1~ : mmap , -1 : normal case \*/

<process>

struct process {

struct list mapping\_list; /\* Manage mmap mappings \*/

void \* stack\_end; /\* end of stack to check over stack\*/

void \* stack\_start; /\* start of stack to check over stack\*/

---- ALGORITHMS ----

**C2: Describe how memory mapped files integrate into your virtual memory subsystem.  Explain how the page fault and eviction processes differ between swap pages and other pages.**

당장 필요하지 않는 page들을 모두배정하는 것이 아니라, 필요할 때에만 부르기 때문에 physical memory를 좀 더 효율적으로 사용할 수 있다.

swap의 경우엔, page fault가 일어나면 disk 에서 swap in한다.

memory mapped file 의 경우엔, eviction이 일어나면 disk에 swap out하지 않고,  원래의 file에 써준다.

**C3: Explain how you determine whether a new file mapping overlaps any existing segment.**

mmap system call 이 발생하면 <mapped file>을 관리하는 mapping\_list와, <executable file의 주소>를 관리하는 file\_list, <stack 주소>를 관리하는 stack\_end, stack\_start 변수를 항상 미리 확인하고 겹치지 않는 경우 mapping을 해준다.

---- RATIONALE ----

**C4: Mappings created with "mmap" have similar semantics to those of data demand-paged from executables, except that "mmap" mappings are written back to their original files, not to swap.  This implies that much of their implementation can be shared. Explain you’re your implementation either does or does not share much of the code for the two situations.**

우리는 같은 file\_table구조를 사용했다. load segment에서 file table entry인 fte에 정보를 채워 process당 load\_file\_table 을 만들어주었고, mmap당 mapping을 만들어 그 안에 file table entry인  fte를 쓰는 file table을 만들었다. 그래서 각 mapping과 process의 load\_segment는 같은 file table구조를 사용한다. 따라서 file table의 구조, 함수를 사용할 수 있다.

그러나 load segment에서 넣어준 entry는 load\_file\_table로 만들어 (mapping\_list 에 넣어주지 않고) 따로 process당 관리해준다.

SURVEY QUESTIONS

================

Answering these questions is optional, but it will help us improve the course in future quarters.  Feel free to tell us anything you want--these questions are just to spur your thoughts.  You may also choose to respond anonymously in the course evaluations at the end of the quarter.

**>> In your opinion, was this assignment or any one of the two problems in it, too easy or too hard?  Did it take too long or too little time?**

**>> Did you find that working on a particular part of the assignment gave you greater insight into some aspect of OS design?**

**>> Is there some particular fact or hint we should give students in future quarters to help them solve the problems?  Conversely, did you find any of our guidance to be misleading?**

**>> Do you have any suggestions for the TAs to more effectively assist students, either for future quarters or the remaining projects?**

**>> Any other comments?**