Project 3: Virtual Memory

1조
권 민철<ventania1680@gmail.com>
김 준영<john5910@naver.com>
이 장원<rex94@naver.com>

---- PRELIMINARIES ----

Contribution
권 민철 : 1

김 준영 : 1

이 장원 : 1

======PAGING======

---- DATA STRUCTURES ----

pagedir.c

pagedir_create (void) : 커널의 가상 주소에 대한 page directory를 생성

pagedir_destroy (uint32_t *pd) : page directory에 할당된 리소스 반

lookup_page (uint32_t *pd, const void *vaddr, bool create) : 해당 가상 주소에 대한 page table entry의 주소를 반환한다. 없으면 생성한다.

pagedir_set_page(uint32_t *pd, void *upage, void *kpage, bool writable) : 사용자 가상 주소에 커널 가상 주소를 맵핑해준다.

pagedir_get_page (uint32_t *pd, const void *uaddr) : 유저 가상 주소에 대한 실제 메모리 주소에 해당하는 커널 가장 주소를 반환한다.

pagedir_clear_page (uint32_t *pd, void *upage) : page에 null pointer 가 없게 clear 해준다.

pagedir_is_dirty (uint32_t *pd, const void *vpage) : 가상 page table entry 이 dirty이면 true를 반환한다.

pagedir_set_dirty (uint32_t *pd, const void *vpage, bool dirty) : page table entry 의 dirty bit 를 설정한다.

pagedir_is_accessed (uint32_t *pd, const void *vpage) : pagedir 이 생성 이후 접근한 적이 있다면 true를 반환한다.

pagedir_activate (uint32_t *pd) : page table의 실제 주소를 page directory register에 저장한다.

active_pd (void) : page directory base register를 page directory로 복사하고 반환한다.

invalidate_pagedir (uint32_t *pd) : page directory 의 TLB bit를 설정한다.

page.c

destory_page() : page 삭제

page_exit() : 현재 쓰레드의 page table 삭제

page_for_addr(address) : address 가 포함이 된 page의 시작 주소를 반환

do_page_in() : 해당 주소의 페이지를 physcial memory로부터 불러온다.

bool page_in() : page fault 가 발생했을 때 fault가 발생한 주소에 있는 page를 page in 해준다.

bool page_out() : dirty bit를 참조를 해서 사용하지 않았다면 그냥 놔두고 사용했다면 swap out을 통해 page out을 한다.

bool page_accessed_recently() : page 사용여부를 확인하고 사용했다면 pte bit 를 true로 바꿔준다.

struct page * page_allocate() : page 테이블에 virtual address 정보를 입력해준다.

void page_deallocate() : 할당을 취소해준다.

unsigned page_hash() : hashing 된 값을 쓸 수 있는 page정보로 바꾼다
bool page_lock() : page 가 안정됐으면 lock()을 건다.
bool page_unlock() : lock()을 푼다.

frame.c

frame_init(void) : 페이지가 저장될 공간들(frames)을 초기화한다.
try_frame_alloc_and_lock(struct page *page) : frame을 탐색해서 빈 공간 있다면 그곳에 page를 넣어주고 없다면 frame중 하나를 선택해서 page out을 해주고 그자리에 데이터를 써준다.
frame_lock() : lock를 걸어서 수정되지 않게 한다.
frame_free() : frame에 할당된 memory를 해제한다.
frame_unlock() : frame lock 을 푼다.

---- ALGORITHMS ----
각 페이지 구조체는 프레임 구조체를 멤버로 가지고 있다. 프레임 구조체는 커널 버츄얼 메모리의 포인터를 가지고 있고, 그것을 가진 페이지에 대한 정보를 갖고 있다. 페이지가 만들어질 때 프레임은 NULL로 설정돼 있고 페이지가 사용될 때 공간이 할당된다.
page table은 hash table 구조로 되어있다.

======SWAP IN/OUT=======

---- DATA STRUCTURES ----
swap_in() : physcial memory에 있는 데이터를 page로 읽어온다.
swap_out() : page의 데이터를 physcial 에 써 준다.

---- ALGORITHMS ----
page fault 가 일어나서 page를 교체해줘야 하는 상황에 page table에 있는 page들을 순서대로 조회하여서 최근 사용하지 않은 page를 swap out 하고 사용할 데이터를 swap in 해준다. 이때 조회한 page가 최근 사용되었다면 reference bit를 다시 초기화해준다.
만약 page table에 있는 모든 page가 reference bit이 true라면 한번 더 search한다.