OS Assignment 9 Report

name：刘乐奇

sid：12011327

Ubuntu用户名：lynchrocket

What happens when the process accesses a memory page not presenting in the physical memory.

When the process accesses a memory page not presenting in the physical memory, a page fault will happen and an interrupt will be triggered. Then operating system will look up the address from PTE and request the disk for the required page. After disk I/O, operating system will update the PTE in the page table with the present bit to be 1 and PFN to be the memory location. And then retry the instruction. The next time the instruction executed, it will successfully access the page in the physical memory.

Realize Clock algorithm

Implementation of `swap_clock.c` is as below.

```c
list_entry_t pra_list_head, *curr_ptr;

static int
_clock_init_mm(struct mm_struct *mm)
{
    //TODO
    list_init(&pra_list_head);
    mm->sm_priv = &pra_list_head;
    return 0;
}

static int
_clock_map_swappable(struct mm_struct *mm, uintptr_t addr, struct Page *page, int swap_in)
{
    //TODO
    list_entry_t *head = (list_entry_t*) mm->sm_priv;
    list_entry_t *entry = &(page->pra_page_link);

    assert(entry != NULL && head != NULL);

    pte_t *ptep = get_pte(mm->pgdir, page->pra_vaddr, 0);
    *ptep |= PTE_A;

    list_add_before(head, entry);

    return 0;
}
```

```c
static int
_clock_swap_out_victim(struct mm_struct *mm, struct Page ** ptr_page, int in_tick)
{
    //TODO
    list_entry_t *head = (list_entry_t*) mm->sm_priv;
    assert(head != NULL);
    assert(in_tick == 0);

    struct Page *victim;
    int cnt = 2;
    while(cnt-- > 0) {
        curr_ptr = head;
        while((curr_ptr = list_next(curr_ptr)) != head) {
            victim = le2page(curr_ptr, pra_page_link);
            pte_t *ptep = get_pte(mm->pgdir, victim->pra_vaddr, 0);
            if (!(*ptep & PTE_A)) {
                *ptr_page = victim;
                list_del(head);
                list_add(curr_ptr, head);
                list_del(curr_ptr);
                mm->sm_priv = head;
                return 0;
            }
            if (*ptep & PTE_A) {
                *ptep &= (~PTE_A);
            }
        }
    }

    *ptr_page = NULL;
    return 0;
}
```

The running result is as below.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
page falut at 0x00003000: K/W
Store/AMO page fault
page falut at 0x00004000: K/W
set up init env for check_swap over!
---------Clock check begin----------
write Virt Page c in clock_check_swap
write Virt Page a in clock_check_swap
write Virt Page d in clock_check_swap
write Virt Page b in clock_check_swap
write Virt Page e in clock_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
write Virt Page b in clock_check_swap
write Virt Page a in clock_check_swap
Store/AMO page fault
page falut at 0x00001000: K/W
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
swap_in: load disk swap entry 2 with swap_page in vadr 0x1000
write Virt Page b in clock_check_swap
write Virt Page c in clock_check_swap
Store/AMO page fault
page falut at 0x00003000: K/W
swap_out: i 0, store page in vaddr 0x4000 to disk swap entry 5
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
write Virt Page d in clock_check_swap
Store/AMO page fault
page falut at 0x00004000: K/W
swap_out: i 0, store page in vaddr 0x5000 to disk swap entry 6
swap_in: load disk swap entry 5 with swap_page in vadr 0x4000
write Virt Page e in clock_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x2000 to disk swap entry 3
swap_in: load disk swap entry 6 with swap_page in vadr 0x5000
write Virt Page a in clock_check_swap
Clock check succeed!
check_swap() succeeded!
```