CS302:

Assignment8 Report

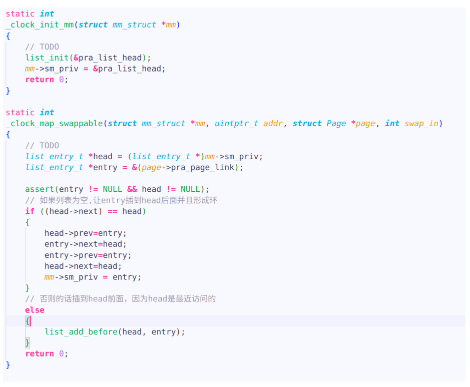
Name: 陆荻芸 SID: 12011537

1 Answer of Question 1

When a process accesses a memory page that is not present in physical memory, a page fault occurs. The page fault exception will be triggered then trapped into kernel to let OS to handle. OS will first go to the **page fault handler** to determine what to do. The handler then check if there are **free** page frame in the physical memory to be allocated. If not, then issue a request to fetch a page from the disk and to **replace** a page in the physical memory according to some page replacement policy. When the disk I/O completes, the OS will then update the page table to mark the page as present, update the PFN of PTE, and retry the instruction returned from the exception handler. After that a TLB miss will happen then the TLB will be updated.

2 Answer of Question 2

The implementation of the functions are shown below.

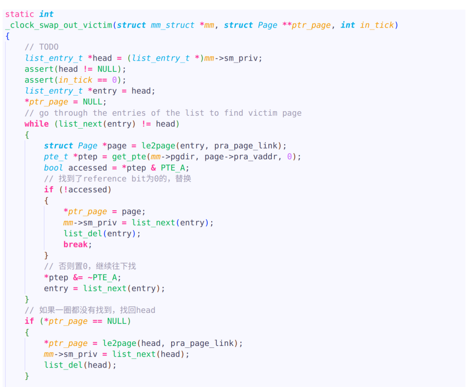


```c
static int
_clock_init_mm(struct mm_struct *mm)
{
    // TODO
    list_init(&pra_list_head);
    mm->sm_priv = &pra_list_head;
    return 0;
}

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);
    // 如果列表为空,让entry插到head后面并且形成环
    if ((head->next) == head)
    {
        head->prev=entry;
        entry->next=head;
        entry->prev=entry;
        head->next=head;
        mm->sm_priv = entry;
    }
    // 否则的话插到head前面，因为head是最近访问的
    else
    {
        list_add_before(head, entry);
    }
    return 0;
}
```

(a) Init function and Swappable Function

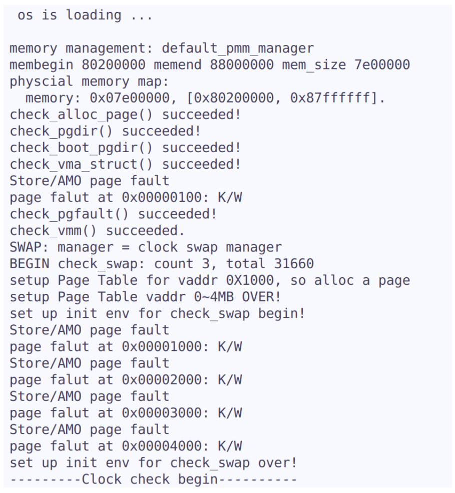


```c
static int
_clock_swap_out_victim(struct mm_struct *mm, struct Page **ptr_page, int in_tick)
{
    // TODO
    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    assert(head != NULL);
    assert(in_tick == 0);
    list_entry_t *entry = head;
    *ptr_page = NULL;
    // go through the entries of the list to find victim page
    while (list_next(entry) != head)
    {
        struct Page *page = le2page(entry, pra_page_link);
        pte_t *ptep = get_pte(mm->pgdir, page->pra_vaddr, 0);
        bool accessed = *ptep & PTE_A;
        // 找到了reference bit为0的，替换
        if (!accessed)
        {
            *ptr_page = page;
            mm->sm_priv = list_next(entry);
            list_del(entry);
            break;
        }
        // 否则置0，继续往下找
        *ptep &= ~PTE_A;
        entry = list_next(entry);
    }
    // 如果一圈都没有找到，找回head
    if (*ptr_page == NULL)
    {
        *ptr_page = le2page(head, pra_page_link);
        mm->sm_priv = list_next(head);
        list_del(head);
    }
```

(b) Swap Out Victim Function

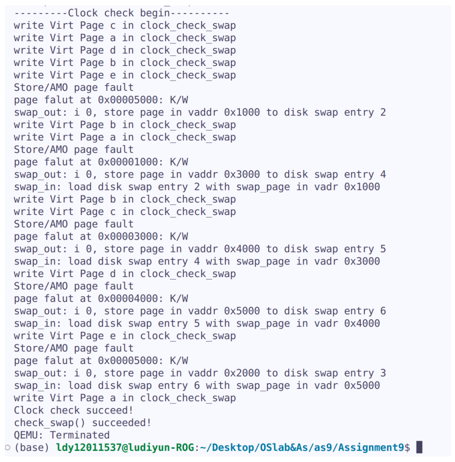
Fig. 1: Implementation

The results are shown below.



```
 os is loading ...

memory management: default_pmm_manager
membegin 80200000 memend 88000000 mem_size 7e00000
physcial memory map:
  memory: 0x07e00000, [0x80200000, 0x87ffffff].
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_vma_struct() succeeded!
Store/AMO page fault
page falut at 0x00000100: K/W
check_pgfault() succeeded!
check_vmm() succeeded.
SWAP: manager = clock swap manager
BEGIN check_swap: count 3, total 31660
setup Page Table for vaddr 0X1000, so alloc a page
setup Page Table vaddr 0~4MB OVER!
set up init env for check_swap begin!
Store/AMO page fault
page falut at 0x00001000: K/W
Store/AMO page fault
page falut at 0x00002000: K/W
Store/AMO page fault
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----------
```



```
---------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!
QEMU: Terminated
○ (base) ldy12011537@ludiyun-ROG:~/Desktop/OSlab&As/as9/Assignment9$
```

Fig. 2: Final Results