操作系统课程实验
Lab3：虚拟内存管理

陈　渝

清华大学计算机系

2014年秋季

大纲

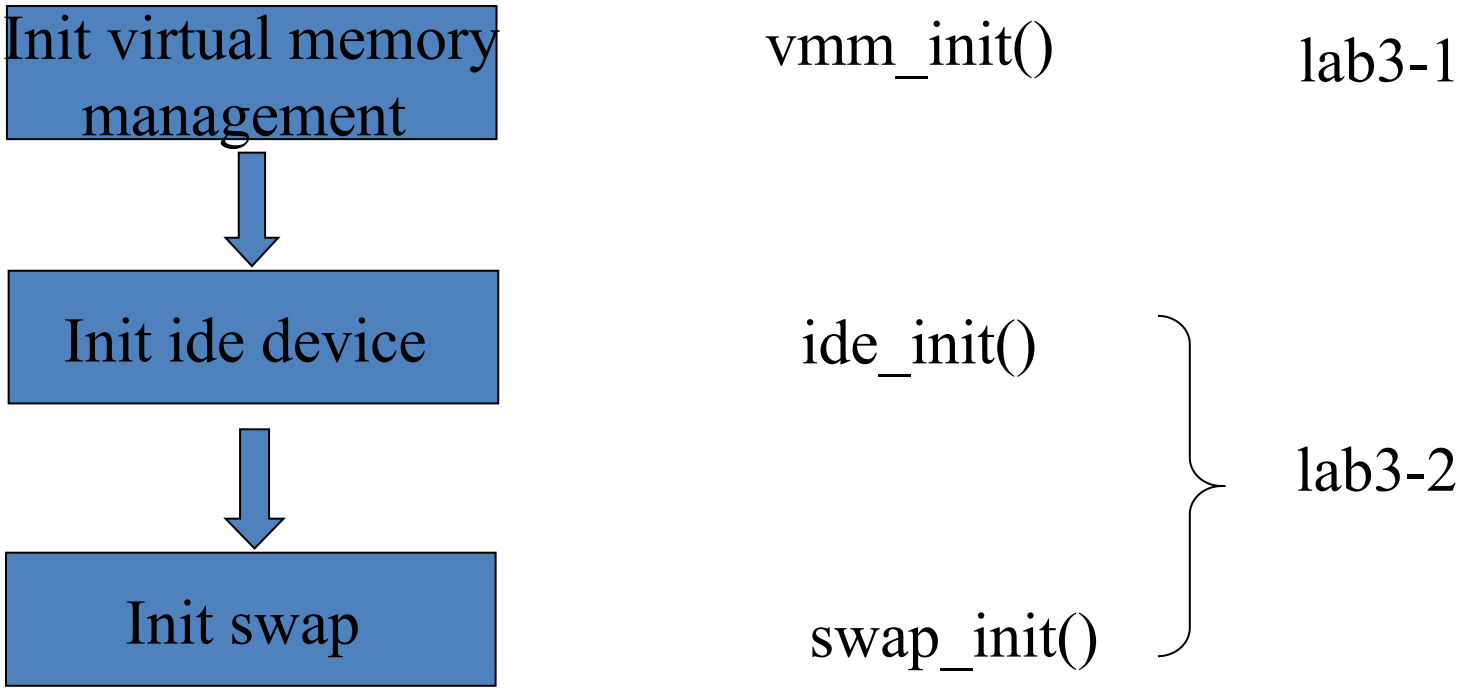
- 处理流程，关键数据结构和功能
- 页访问异常（Page Fault）
- 页换入换出机制（Page Swap Mechanism）

处理流程，关键数据结构和功能

- 虚拟内存管理初始化 (\kern\init\init.c)
  - pmm_init().
  - pic_init(),
  - idt_init().

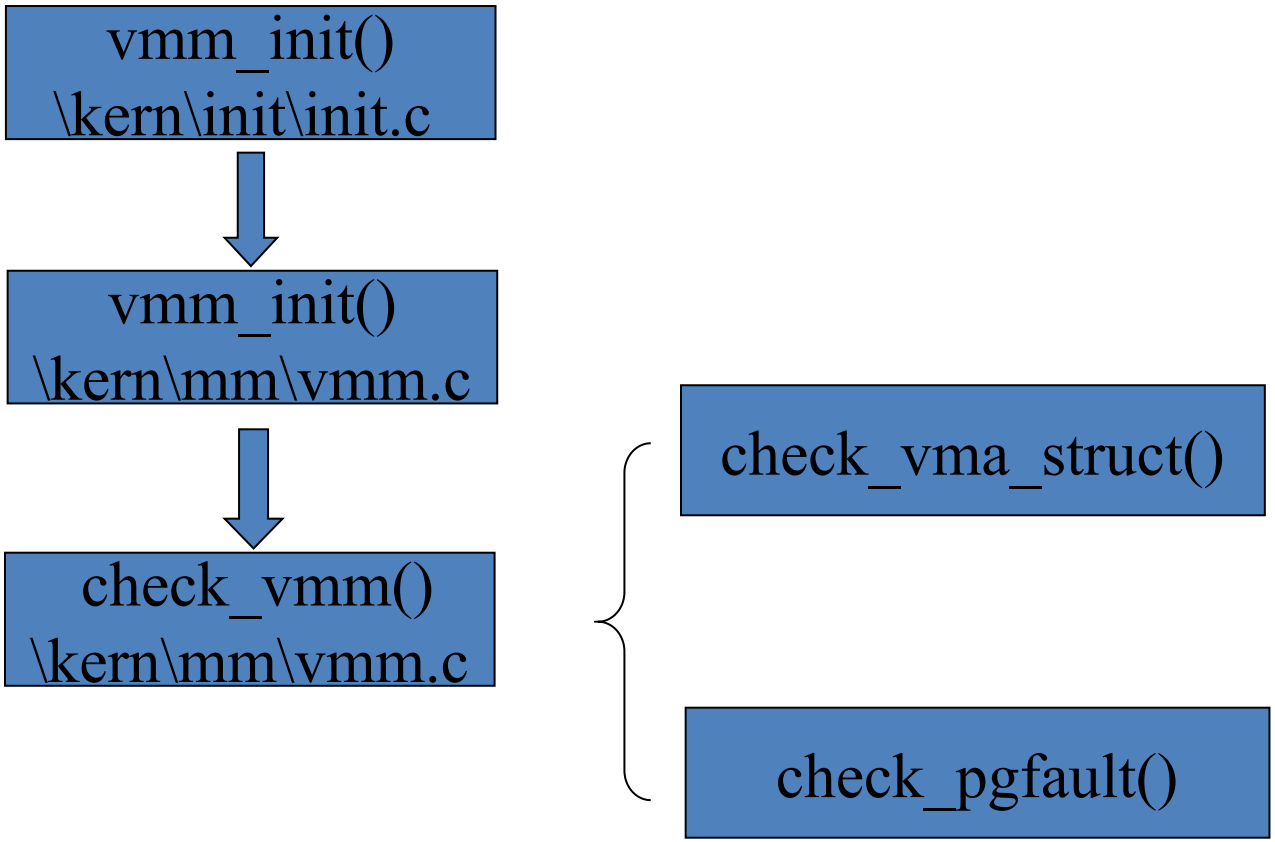
处理流程，关键数据结构和功能

- 虚拟内存管理初始化(\kern\init\init.c)
  - vmm_init()、ide_init() and swap_init()



处理流程，关键数据结构和功能

- 虚拟内存管理初始化(\kern\mm\vmm.c)



处理流程，关键数据结构和功能

- 关键数据结构 (\kern\mm\vmm.h)

```
struct vma_struct {
        // the set of vma using the same PDT
        struct mm_struct *vm_mm;
        uintptr_t vm_start;                     // start addr of vma
        uintptr_t vm_end;                       // end addr of vma
        uint32_t vm_flags;                      // flags of vma
        //linear list link which sorted by start addr of vma
        list_entry_t list_link;
};
```

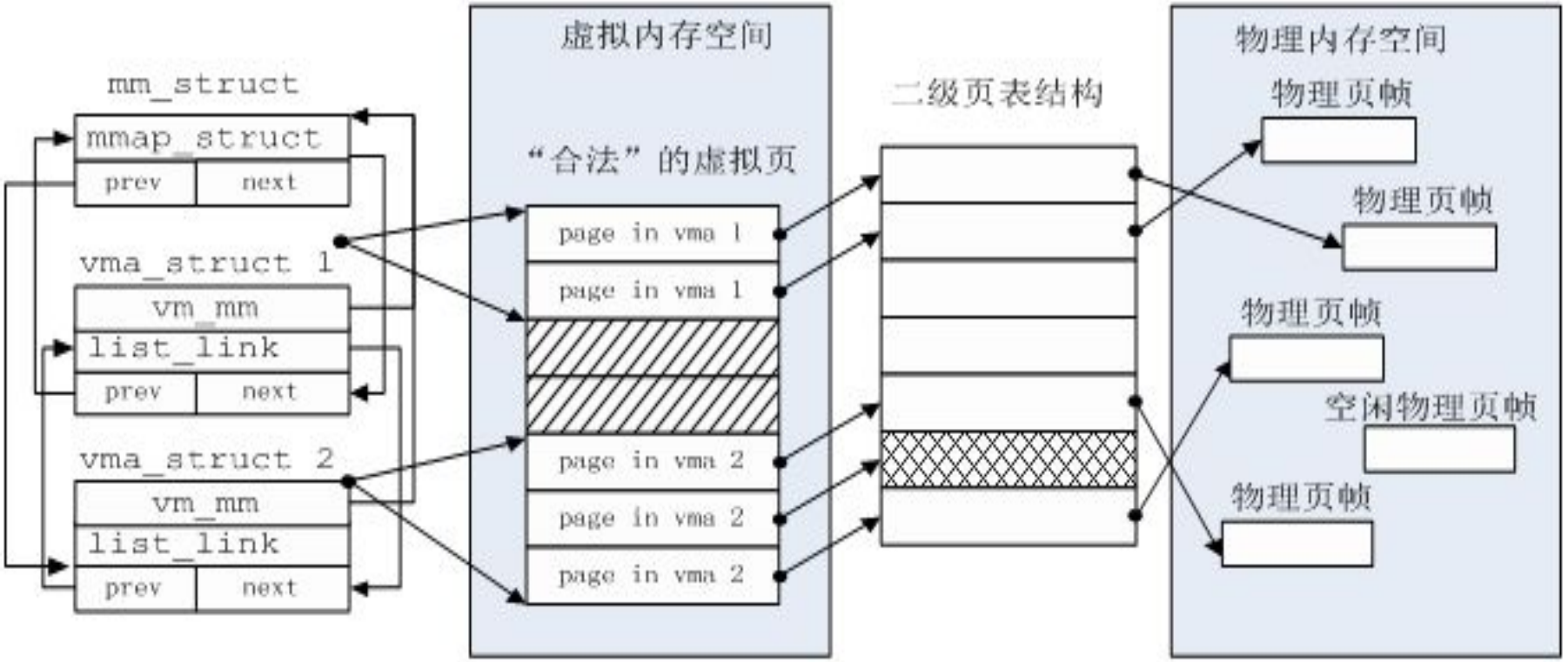
处理流程，关键数据结构和功能

- 关键数据结构 (\kern\mm\vmm.h)

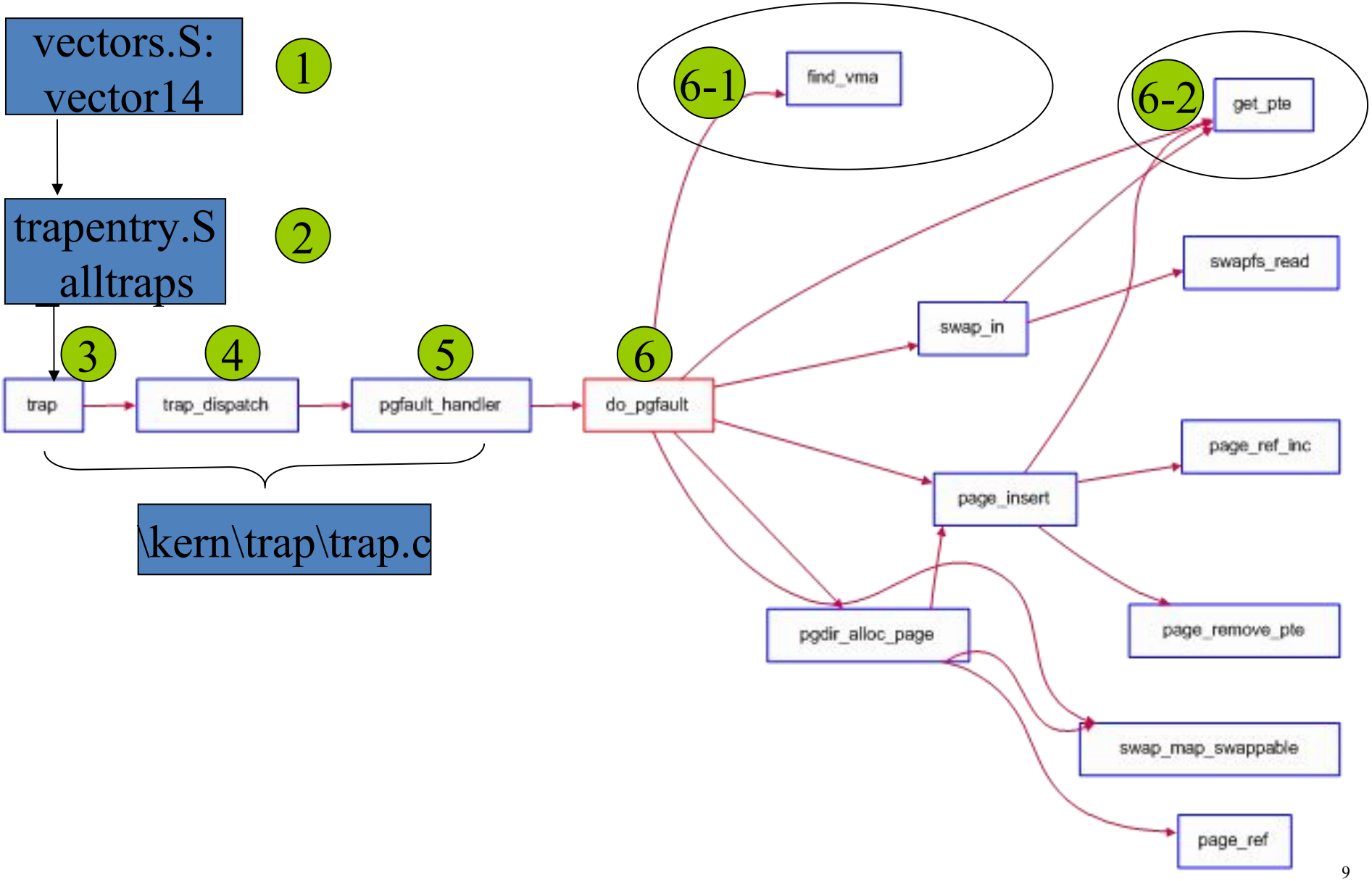
```
struct mm_struct {
        // linear list link which sorted by start addr of vma
        list_entry_t mmap_list;
        // current accessed vma, used for speed purpose
        struct vma_struct *mmap_cache;
        pde_t *pgdir;  // the PDT of these vma
        int map_count; // the count of these vma
        void *sm_priv;           // the private data for swap manager
};
```

处理流程，关键数据结构和功能

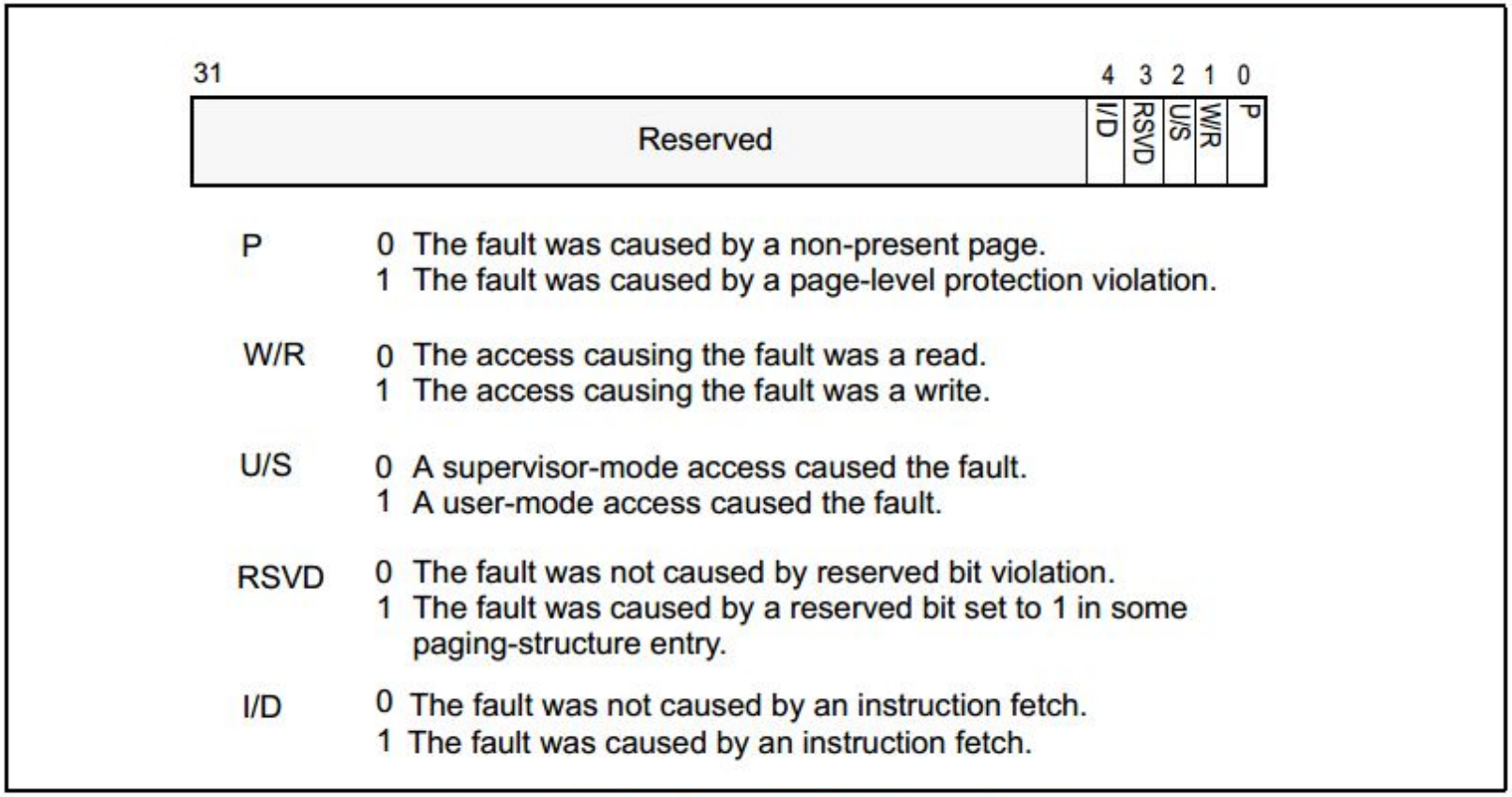
- 关键数据结构（\kern\mm\vmm.h)



页访问异常


vectors.S:
vector14
1
trapentry.S
alltraps
2
3
4
5
6
trap
trap_dispatch
pgfault_handler
do_pgfault
\kern\trap\trap.c
6-1
find_vma
6-2
get_pte
swap_in
swapfs_read
page_ref_inc
page_insert
page_remove_pte
pgdir_alloc_page
swap_map_swappable
page_ref

页访问异常



P	0	The fault was caused by a non-present page.
	1	The fault was caused by a page-level protection violation.
W/R	0	The access causing the fault was a read.
	1	The access causing the fault was a write.
U/S	0	A supervisor-mode access caused the fault.
	1	A user-mode access caused the fault.
RSVD	0	The fault was not caused by reserved bit violation.
	1	The fault was caused by a reserved bit set to 1 in some paging-structure entry.
I/D	0	The fault was not caused by an instruction fetch.
	1	The fault was caused by an instruction fetch.

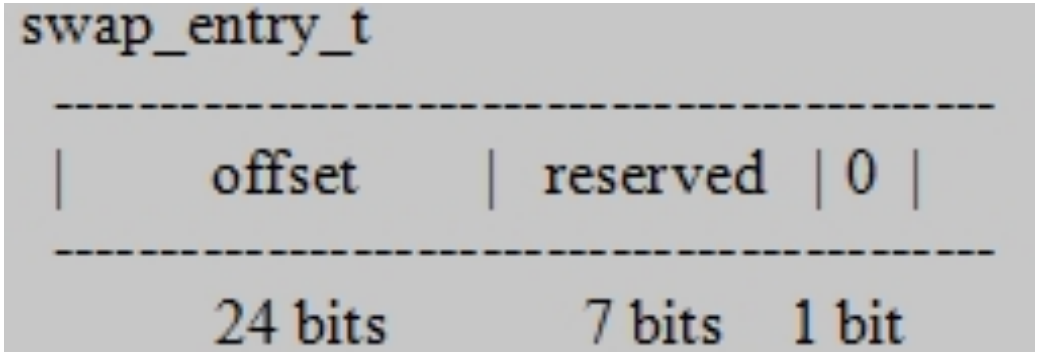
Figure 4-12. Page-Fault Error Code

页换入换出机制

- 需要考虑的问题
  - 应该换出哪个页？
  - 如何建立虚拟页和磁盘扇区的对应关系？
  - 何时进行页换入和换出？
  - 如何设计数据结构支持页替换算法？
  - 怎样进行页换入和换出？

页换入换出机制

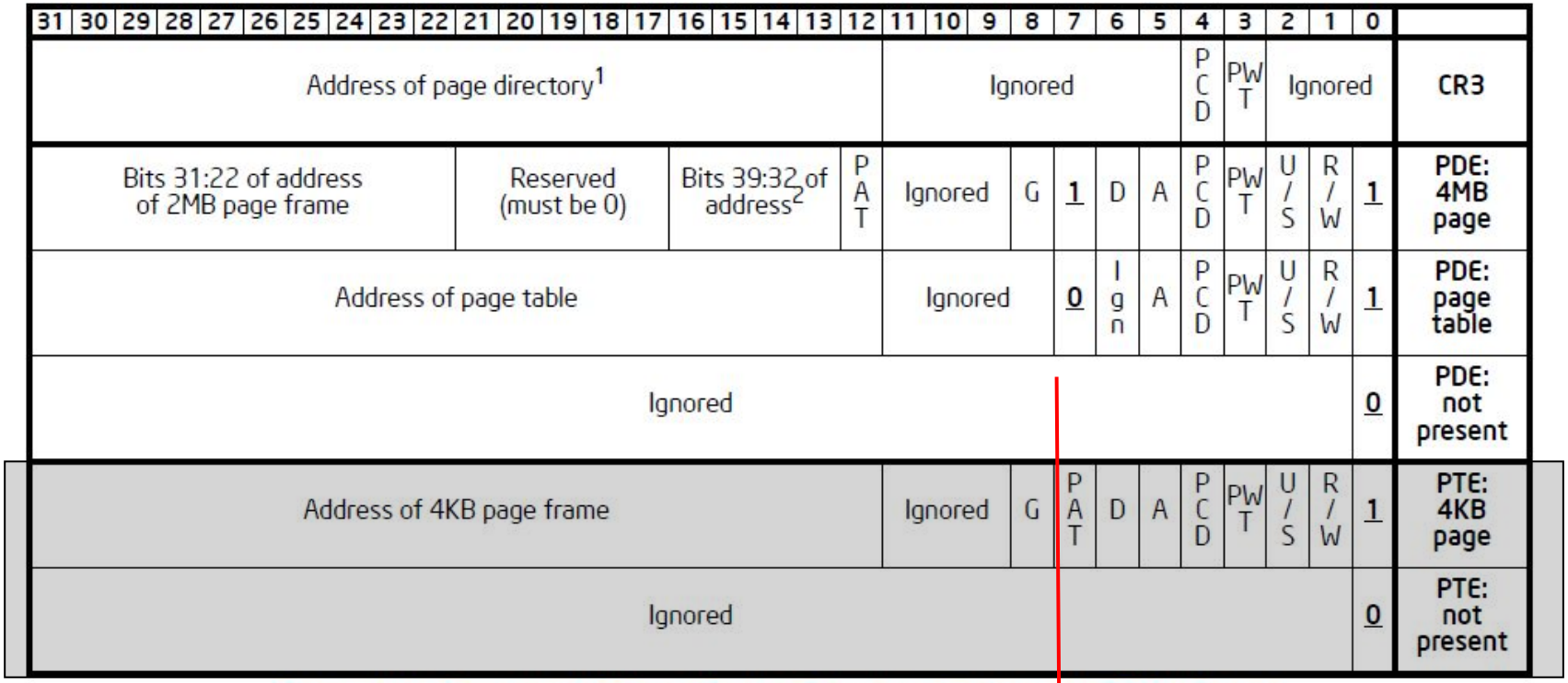
- 应该换出哪个页？
  - (\kern\mm\swap.c)
    - Lab3: check_swap()
- 如何建立虚拟页和磁盘扇区的对应关系？
  - PTE



页换入换出机制

- 页替换算法
  - FIFO: First In First Out (Lab3-2)
  - Clock
  - Enhanced Clock

页换入换出机制



31–12	11–9	8	7	6	5	4	3	2	1	0	
Address of page directory[1]	Ignored					PCD	PWT	Ignored			CR3
Bits 31:22 of address of 2MB page frame; Reserved (must be 0); Bits 39:32 of address[2]; PAT	Ignored	G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 4MB page
Address of page table	Ignored		0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored										0	PDE: not present
Address of 4KB page frame	Ignored	G	PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored										0	PTE: not present

Figure 4-4. Formats of CR3 and Paging-Structure Entries with 32-Bit Paging

- R/W: 1 if this page is writable
- U/S: 1 if this page is accessible in ring 3
- A: 1 if this page has been accessed
- D: 1 if this page has been written
- You may ignore others for now

页换入换出机制

- 何时进行页换入和换出？
  - 换入（Swap in）: (kern\mm\swap.c)

    check_swap() ⇒ page fault ⇒ do_pgfault()

  - 换出（Swap out）
    - 主动策略
    - 被动策略 (ucore)

页换入换出机制

- 如何设计数据结构支持页替换算法？

```
struct Page {
    ......
    list_entry_t    pra_page_link;
    uintptr_t       pra_vaddr;
    };
```

页换入换出机制

- 如何设计数据结构支持页替换算法？ (kern\mm\swap.h, kern\mm\swap_fifo.c)

```
struct swap_manager{
                    ...
/* Called when map a swappable page into the mm_struct */
int (*map_swappable)   (struct mm_struct *mm, uintptr_t addr, struct
Page *page, int swap_in);
                    ...
/* Try to swap out a page, return then victim */
int (*swap_out_victim) (struct mm_struct *mm, struct Page **ptr_page,
int in_tick);
/* check the page relpacement algorithm */
int (*check_swap)(void);
};
```

页换入换出机制

- 怎样进行页换入和换出？
  - swap_check(): kern/mm/swap.c
    - mm_create() : mm; vma_create(): vma; set access range: 4KB-24KB.
    - free_page: simulate 4 physical pages, and 5 continuous virtual pages accessing operation.
    - Set pgfault_num=0, execute check_content_set().
    - Check validity
    - check_content_access(), _fifo_check_swap().

谢谢！