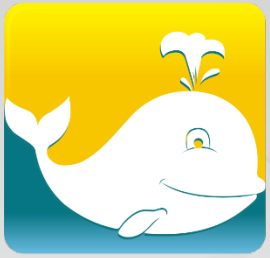


操作系统
Operating Systems

Lec10 实验三 虚拟内存管理

清华大学计算机系

大 纲

- 实验目标：虚存管理
- 回顾历史： lab1 和 lab2
- 了解当下： lab3
- 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 ）

实验目标：虚存管理

- 本次实验是在实验一、二的基础上，
- 借助于页表机制和中断异常处理机制

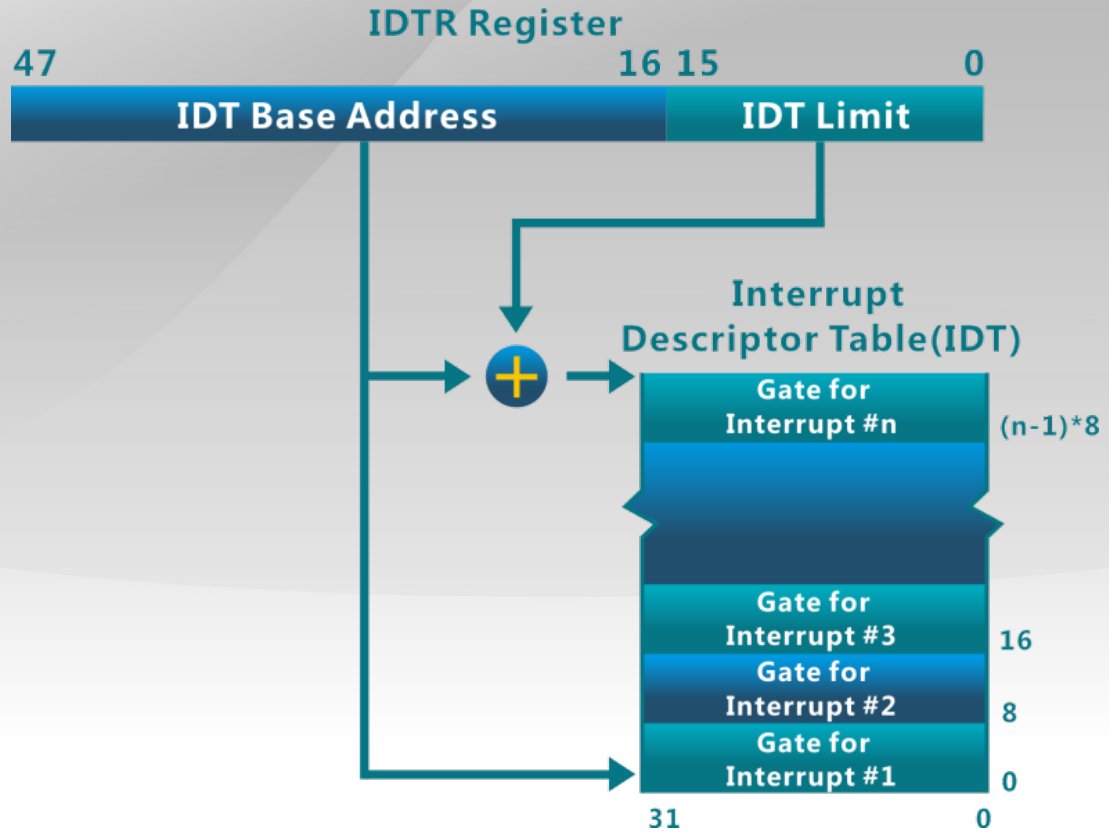
实验目标：虚存管理

- 本次实验是在实验一、二的基础上，
- 借助于页表机制和中断异常处理机制
- 完成 Page Fault 异常处理和 FIFO 页替换算法的实现
- 结合磁盘提供的缓存空间，从而能够支持虚存管理
- 提供一个比实际物理内存空间“更大”的虚拟内存空间

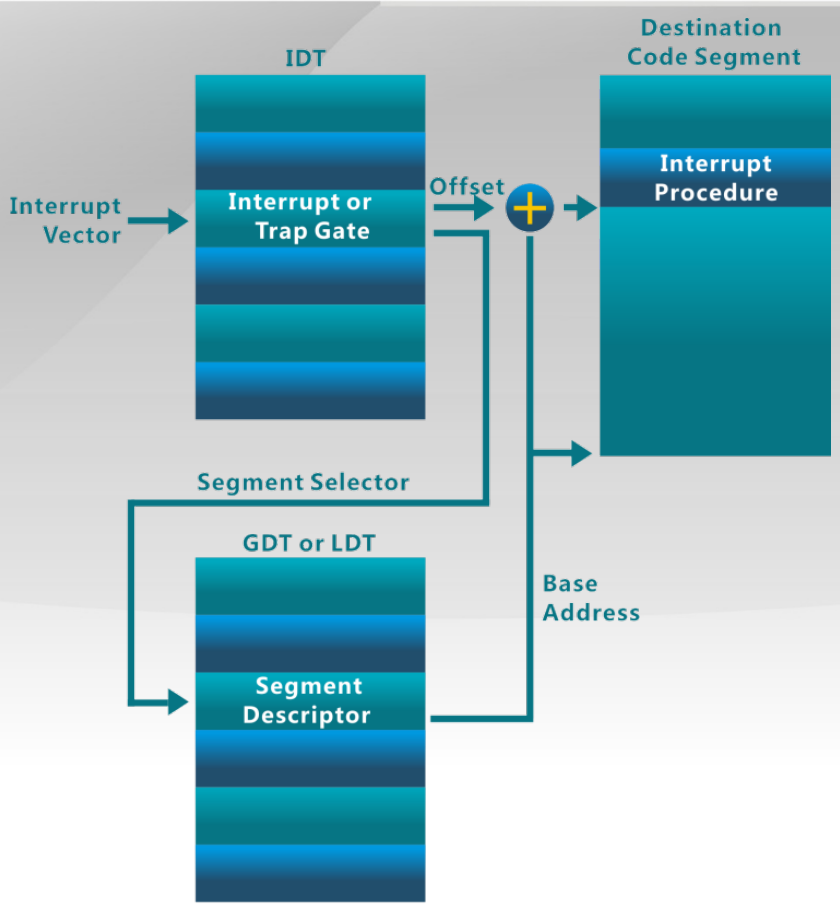
回顾历史：lab1 和 lab2

- Lab1
  - 完成了保护模式和段机制的建立
  - 完成了中断机制的建立
  - 可以输出字符串

回顾历史：lab1 和 lab2



回顾历史：lab1 和 lab2



回顾历史：lab1 和 lab2

- Lab2
  - 查找了内存物理空间
  - 建立了基于连续物理内存空间的动态内存分配与释放机制
  - 完成了页机制的建立

回顾历史：lab1 和 lab2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	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

页目录表基址寄存器 CR3 ，页目录结构以及页表结构

了解当下：lab3

- Lab3：完成虚存管理
  - 建立在 lab1 和 lab2 的基础之上
  - 思考虚存管理总体框架
  - 建立处理页访问错误的异常 / 中断服务例程
  - 实现对硬盘 swap 分区的读写
  - 完成页替换算法
  - 基于上诉实现和对页表的处理，完成支持页替换的虚存管理

了解当下：lab3

- 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理

了解当下：lab3

- 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理
2. 设置虚拟页空间和物理页帧空间，表述不在物理内存中的“合法”虚拟页

了解当下：lab3

- 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理
2. 设置虚拟页空间和物理页帧空间，表述不在物理内存中的“合法”虚拟页
3. 完善建立页表映射、页访问异常处理操作等函数实现

了解当下：lab3

■ 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理
2. 设置虚拟页空间和物理页帧空间，表述不在物理内存中的“合法”虚拟页
3. 完善建立页表映射、页访问异常处理操作等函数实现
4. 执行访存测试，查看建立的页表项是否能够正确完成虚实地址映射

了解当下：lab3

■ 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理
2. 设置虚拟页空间和物理页帧空间，表述不在物理内存中的“合法”虚拟页
3. 完善建立页表映射、页访问异常处理操作等函数实现
4. 执行访存测试，查看建立的页表项是否能够正确完成虚实地址映射
5. 执行访存测试，查看是否正确描述了虚拟内存页在物理内存中还是在硬盘上

了解当下：lab3

■ 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理
2. 设置虚拟页空间和物理页帧空间，表述不在物理内存中的“合法”虚拟页
3. 完善建立页表映射、页访问异常处理操作等函数实现
4. 执行访存测试，查看建立的页表项是否能够正确完成虚实地址映射
5. 执行访存测试，查看是否正确描述了虚拟内存页在物理内存中还是在硬盘上
6. 执行访存测试，查看是否能够正确把虚拟内存页在物理内存和硬盘之间进行传递

了解当下：lab3

■ 虚存管理总体框架

1. 完成初始化虚拟内存管理机制： IDE 硬盘读写，缺页异常处理
2. 设置虚拟页空间和物理页帧空间，表述不在物理内存中的“合法”虚拟页
3. 完善建立页表映射、页访问异常处理操作等函数实现
4. 执行访存测试，查看建立的页表项是否能够正确完成虚实地址映射
5. 执行访存测试，查看是否正确描述了虚拟内存页在物理内存中还是在硬盘上
6. 执行访存测试，查看是否能够正确把虚拟内存页在物理内存和硬盘之间进行传递
7. 执行访存测试，查看是否正确实现了页面替换算法等

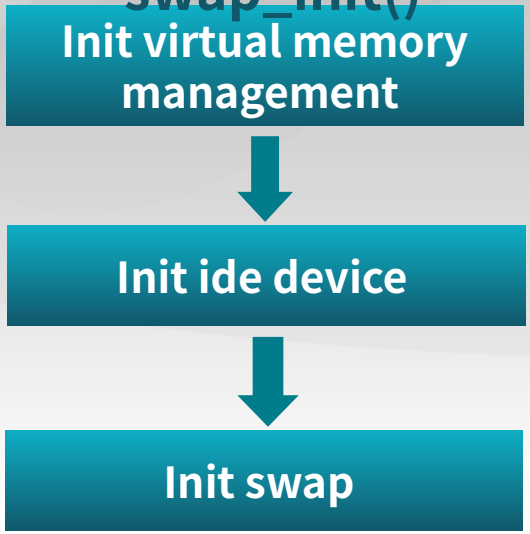
处理流程，关键数据结构和功能

- 虚拟内存管理初始化前（\kern\init\init.c）

pmm_init().	lab2
pic_init(),	lab1
idt_init().	lab1

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()



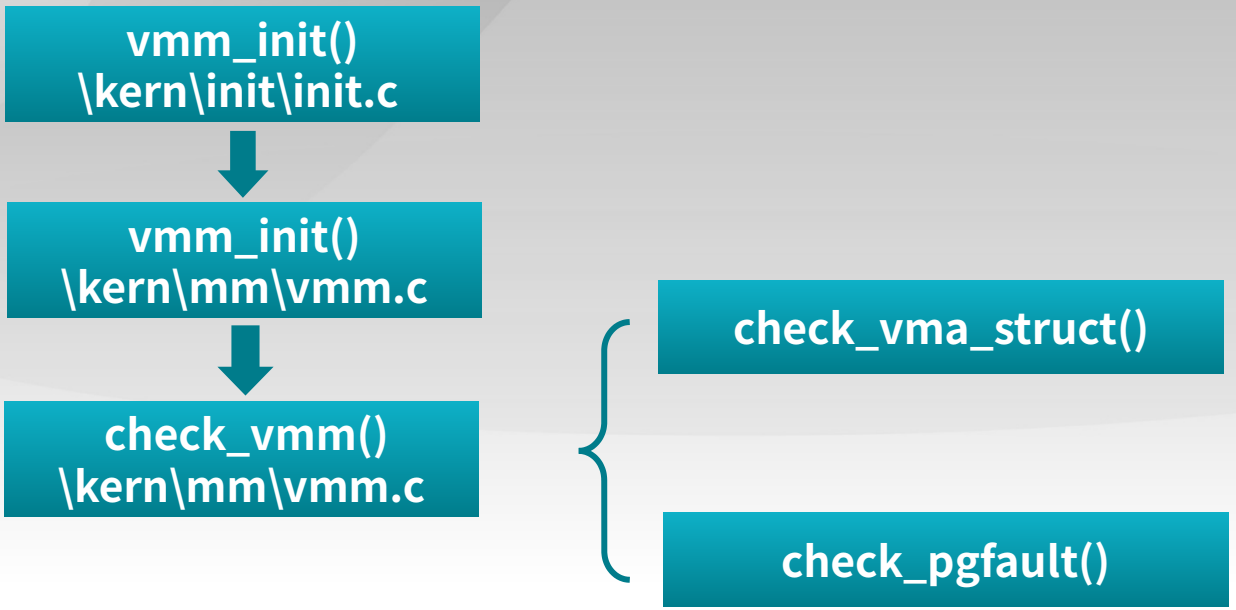
vmm_init()　lab3-1

ide_init()

swap_init()　lab3-2

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

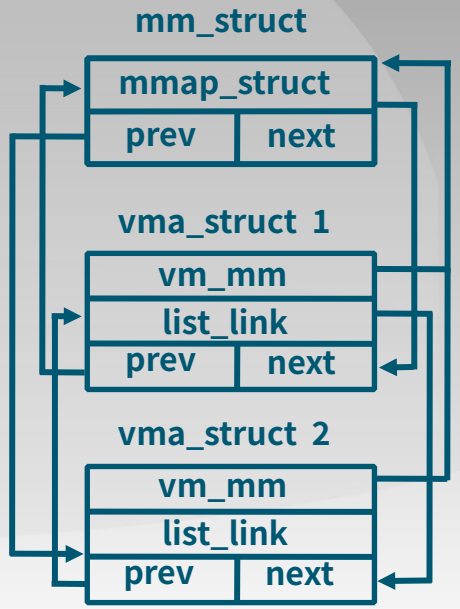
■ 关键数据结构 (\kern\mm\vmm.h)

```
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

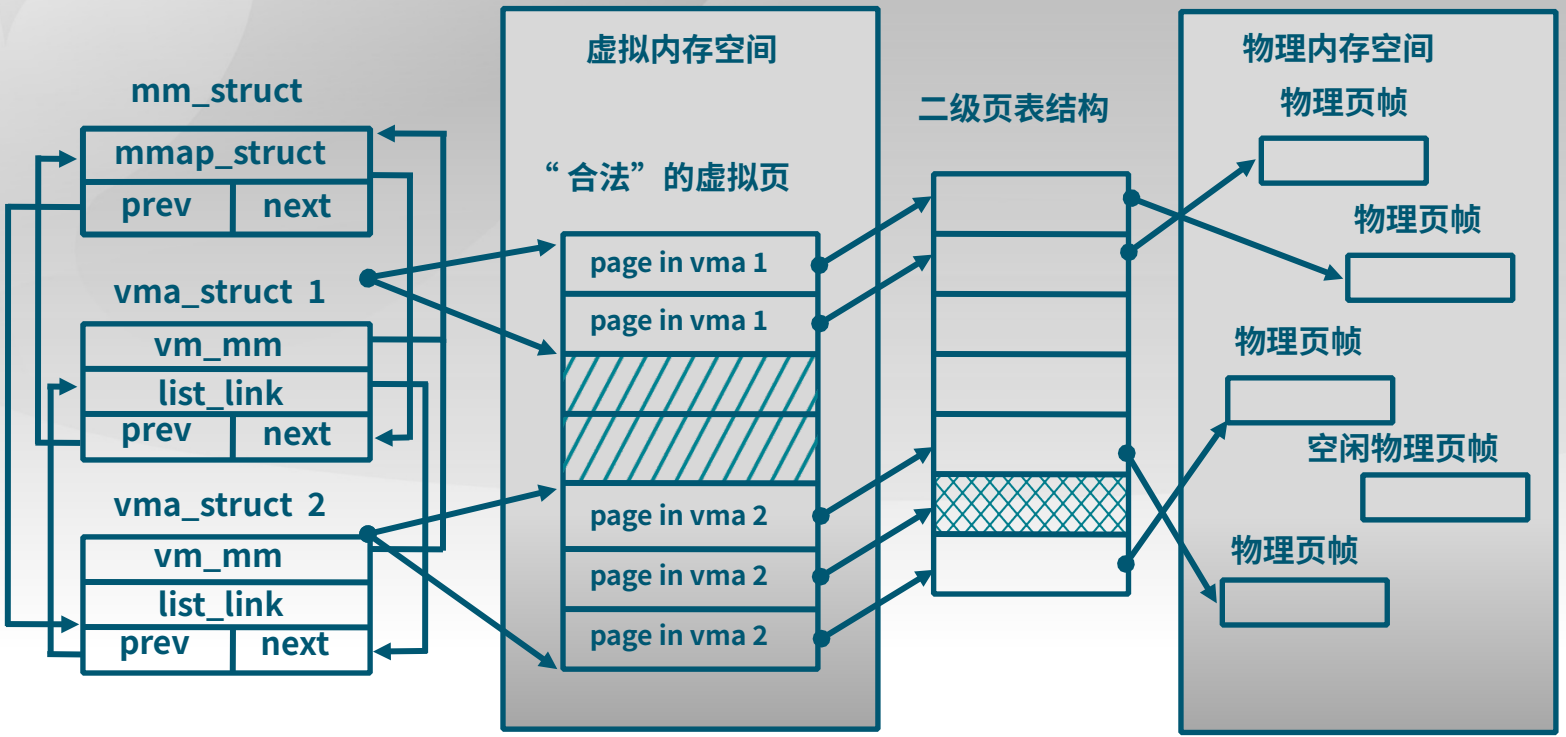
■ 关键数据结构 (\kern\mm\vmm.h)

```
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
};
```

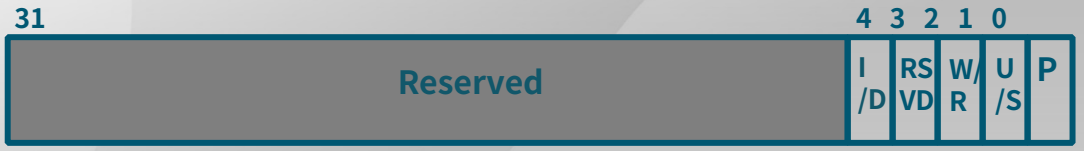


处理流程，关键数据结构和功能

■ 关键数据结构 (\kern\mm\vmm.h)



页访问异常

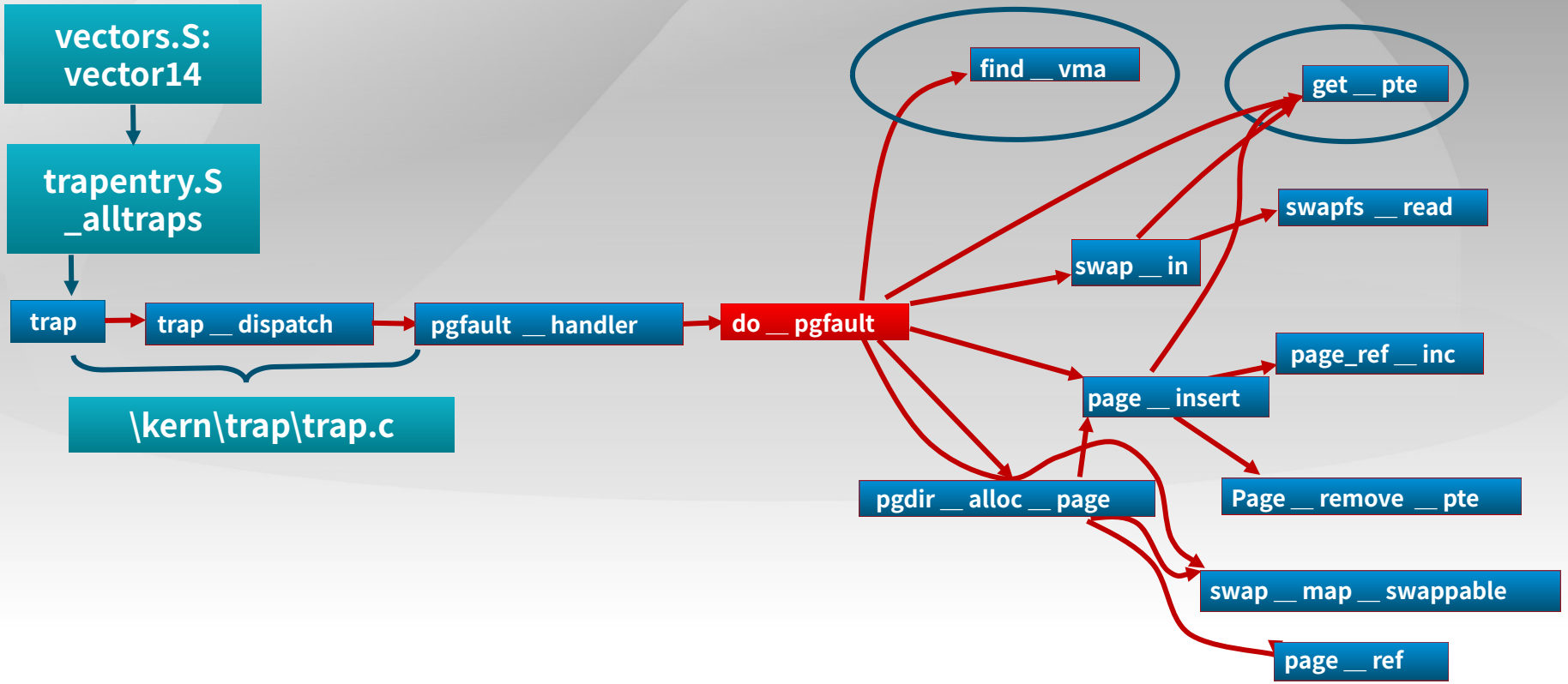


P	0 The fault was caused by a non-present page. 1 The fault was caused by a page-level protection violation.
W/R	0 The access causing the fault was a read. 1 The access causing the fault was a write.
U/S	0 A supervisor-mode access caused the fault. 1 A user-mode access caused the fault.
RSVD	0 The fault was not caused by reserved bit violation. 1 The fault was caused by a reserved bit set to 1 in some paging-structure entry. When PSE/PAE flags in CR4 are set to 1.
I/D	0 The fault was not caused by an instruction fetch. 1 The fault was caused by an instruction fetch.

The CR2 register contains the 32-bit linear address that caused the fault.

Figure 4 – 12. Page-Fault Error Code

页访问异常

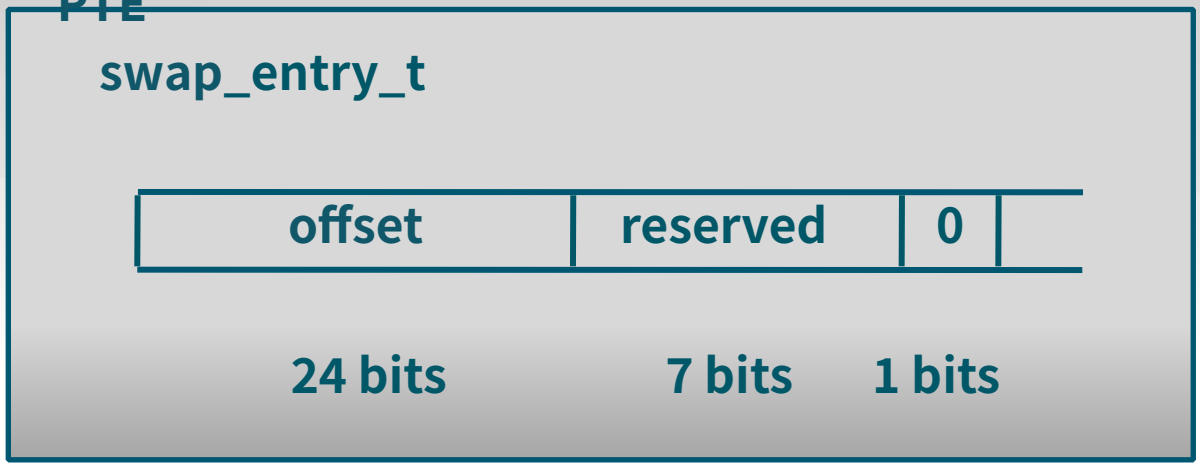

vectors.S:
vector14
trapentry.S
_alltraps
trap
trap __ dispatch
pgfault __ handler
do __ pgfault
\kern\trap\trap.c
find __ vma
get __ pte
swapfs __ read
swap __ in
page_ref __ inc
page __ insert
pgdir __ alloc __ page
Page __ remove __ pte
swap __ map __ swappable
page __ ref

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:
  - 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
  - (LabB-2)
  - ock Enhanced Clock

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)

谢谢！