

操作系统

Operating Systems

L24 内存换入-请求调页

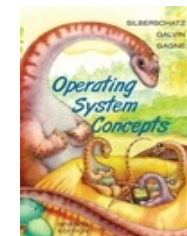
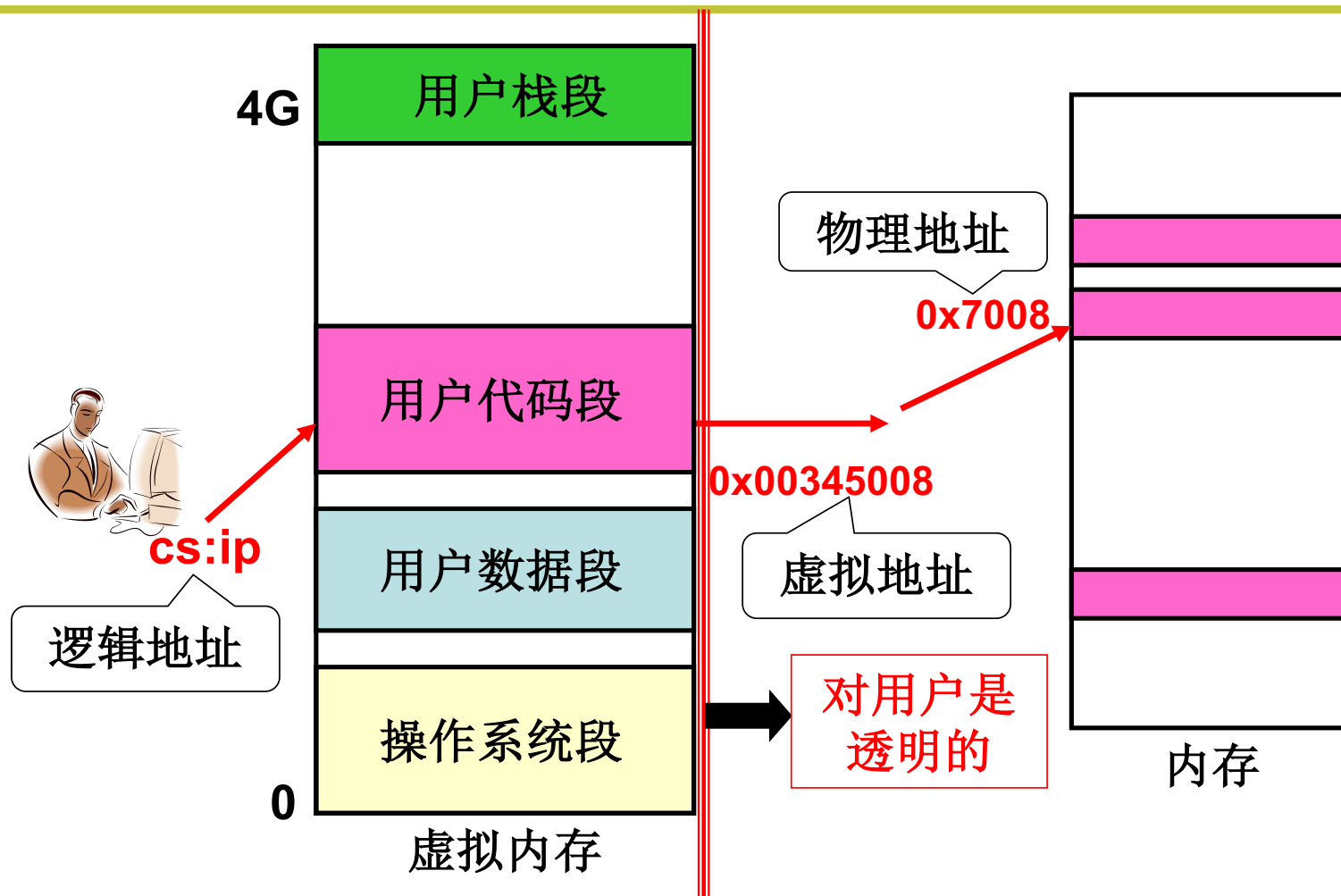
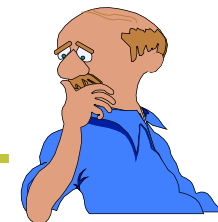
Swap in

lizhijun_os@hit.edu.cn

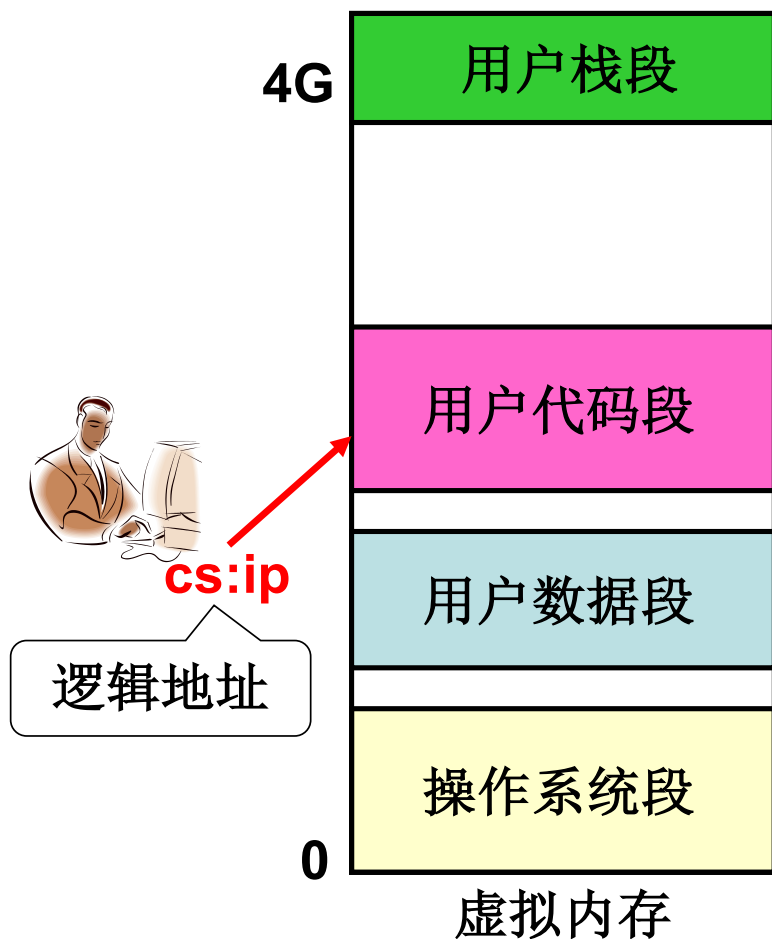
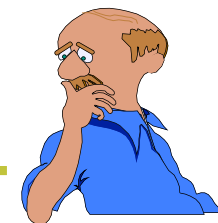
综合楼411室

授课教师：李治军

段、页同时存在

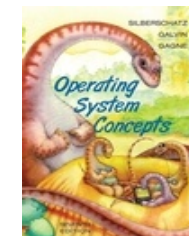


用户眼里的内存!



- **4GB**(大且规整)的“内存”，可供用户使用，如**char *p, p=3G**，实际上就是用地址
- 用户可随意使用该“内存”，就象单独拥有**4G**内存
- 这个“内存”怎么映射到物理内存，用户全然不知

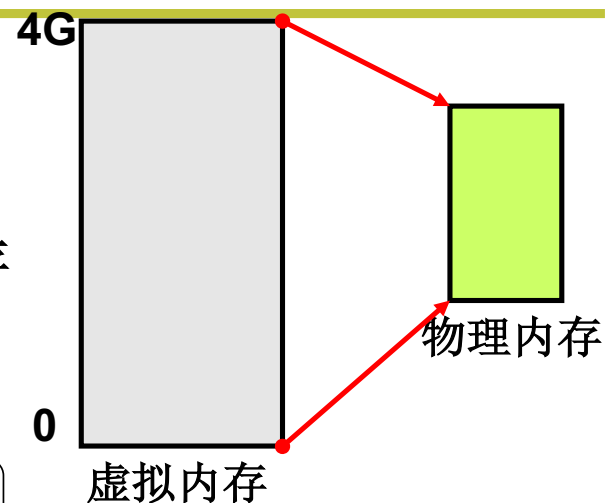
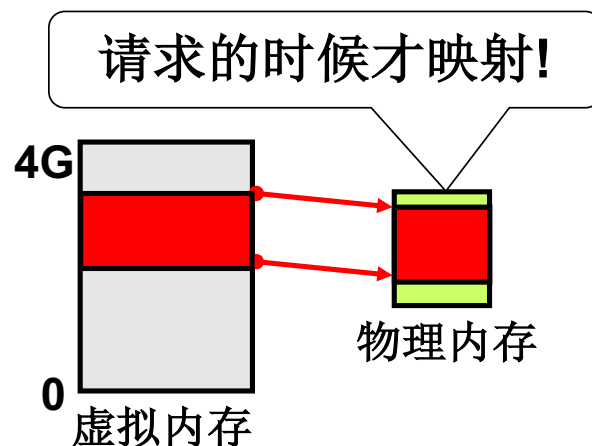
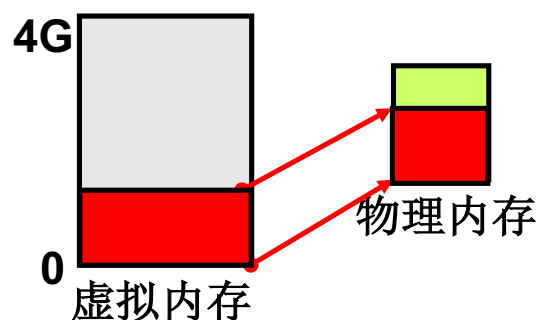
必须映射，否则
不能用!



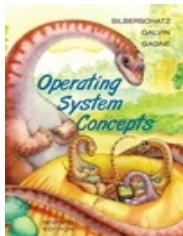
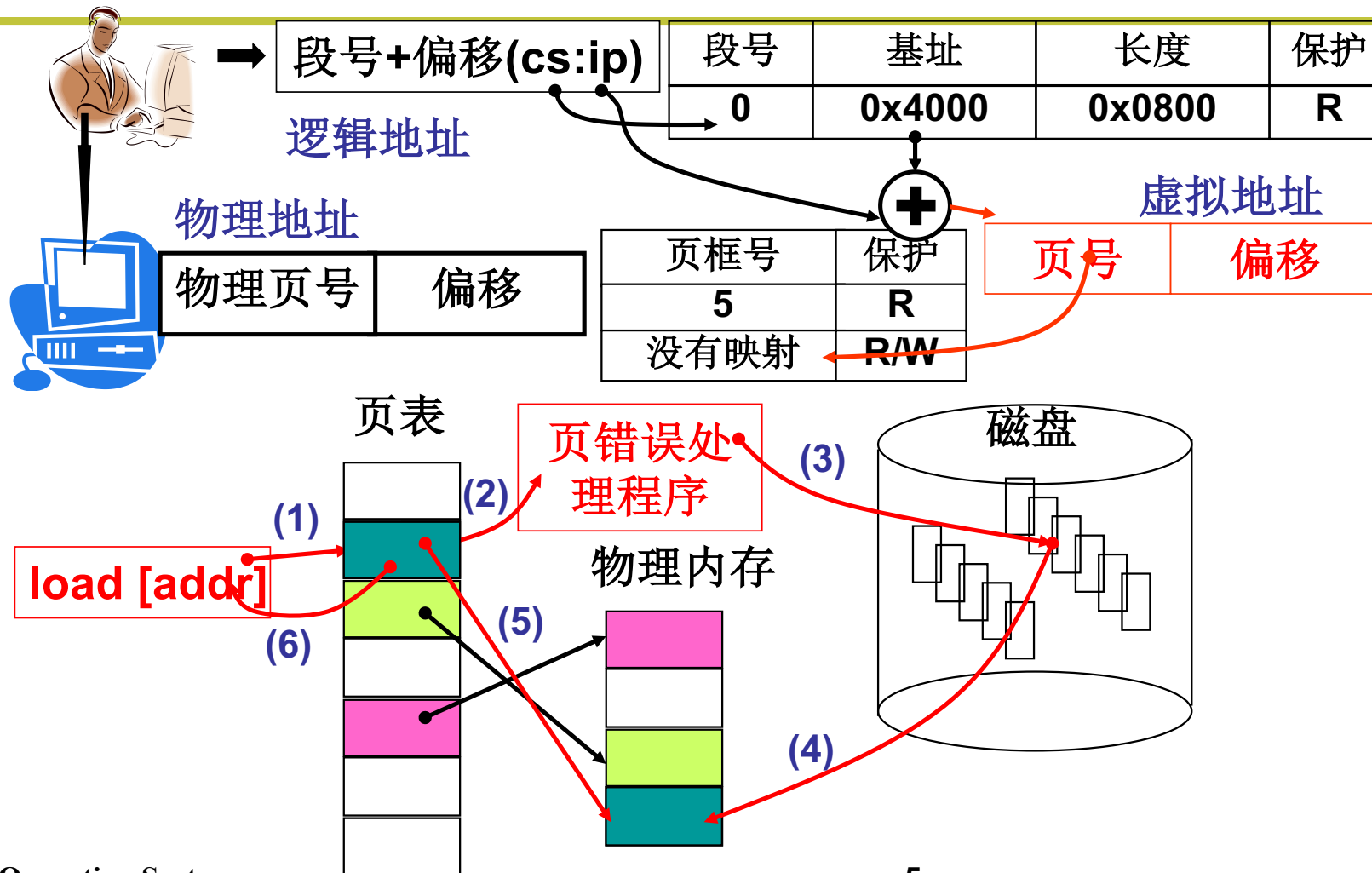
用换入、换出实现“大内存”

■ 左边4G，右边1G怎么办？

- 访问 $p(=0G - 1G)$ 时，将这部分映射到物理内存
- 再访问 $p(=3G-4G)$ 时，再映射这一部分

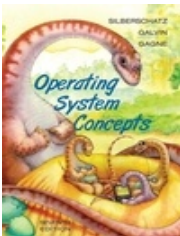


请求调页



问题：采用请求调页而不是请求调段，是因为?()

- A.** 请求调页对用户更透明
- B.** 用户程序需要因为请求调段而重写
- C.** 请求调页的粒度更细，更能提高内存效率
- D.** 请求调段不工作在内核态



一个实际系统的请求调页

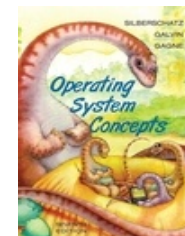
■ 这个故事从哪里开始？

■ 请求调页，当然从缺页中断开始

中断号	名称	说明
12	Segment not Present	描述符所指的段不存在
14	Page fault	页不在内存

```
void trap_init(void)
{ set_trap_gate(14, &page_fault); }
```

```
#define set_trap_gate(n, addr) \
    _set_gate(&idt[n], 15, 0, addr);
```



处理中断page fault

//在linux/mm/page.s中

```
.globl _page_fault
    xchgl %eax, (%esp)
    pushl %ecx
    pushl %edx
    push %ds
    push %es
    push %fs
    movl $0x10, %edx
    mov %dx, %ds
    mov %dx, %es
    mov %dx, %fs
    movl %cr2, %edx
```

错误码被压
到了栈中

页错误线性地址

pushl %edx

pushl %eax

压入参数

testl \$1, %eax

jne 1f

测试标志P

call _do_no_page

jmp 2f

1: call _do_wp_page //保护

2: add \$8, %esp

pop %fs

pop %es

pop %ds

pop %edx

pop %ecx

pop %eax

iret

- 8 -



do_no_page

//在linux/mm/memory.c中

```
void do_no_page(unsigned long error_code,  
                unsigned long address)
```

```
{  address&=0xfffff000;    //页面地址
```

去除偏移量，获取虚拟页号

不是代码和数据!

```
    tmp=address-current->start_code; //页面对应的偏移
```

```
    if(!current->executable||tmp>=current->end_data){
```

```
        get_empty_page(address); return; }
```

```
    page=get_free_page(); 申请物理内存页磁
```

```
    bread_page(page, current->executable->i_dev, nr);
```

将磁盘中的虚拟页放入刚申请的物理页page

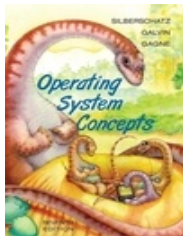
```
    put_page(page, address);该函数用于  
                                修改页表
```

读文件系统...

```
void get_empty_page(unsigned long address)
```

```
{  unsigned long tmp = get_free_page();
```

```
    put_page(tmp, address);}
```



put_page

//在linux/mm/memory.c中

```
unsigned long put_page(unsigned long page, //物理地址
                        unsigned long address)
{ unsigned long tmp, *page_table;
  page_table=(unsigned long *) ((address>>20)&ffc); 找到缺页的页目录项
  if ((*page_table)&1)
    page_table=(unsigned long*) (0xfffff000&*page_table);
  else{
    tmp=get_free_page();
    *page_table=tmp|7;
    page_table=(unsigned long*) tmp;}
  page_table[(address>>12)&0x3ff] = page|7; 建立映射
  return page;
}
```

页目录项

找到缺页的页目录项

建立映射

