操作系统 Operating Systems

L23 段页结合的实际内存管理

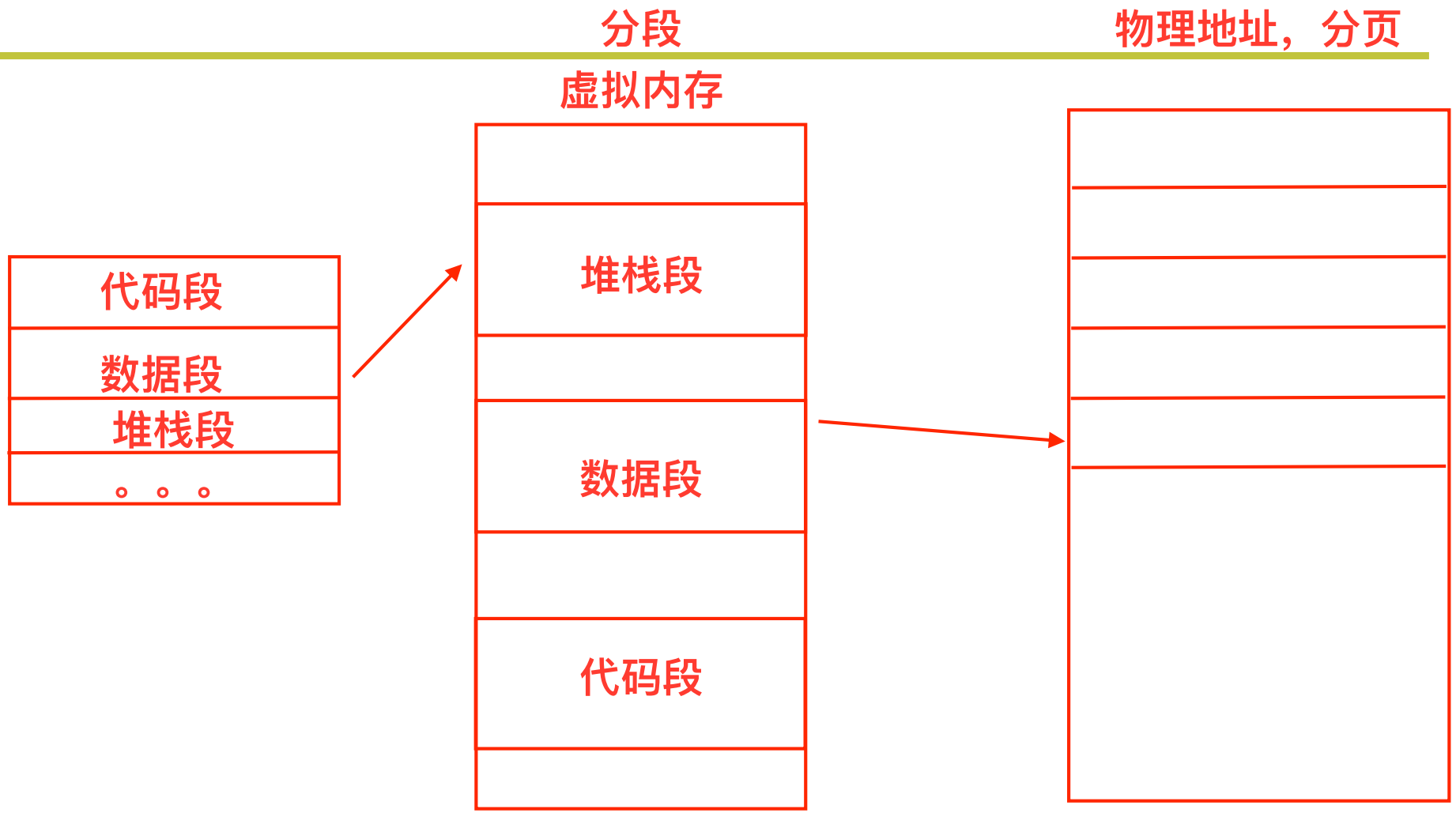
Segmentation & Paging

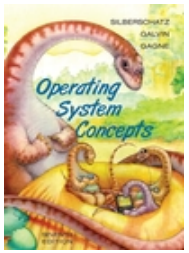
lizhijun_os@hit.edu.cn
综合楼411室

授课教师：李治军

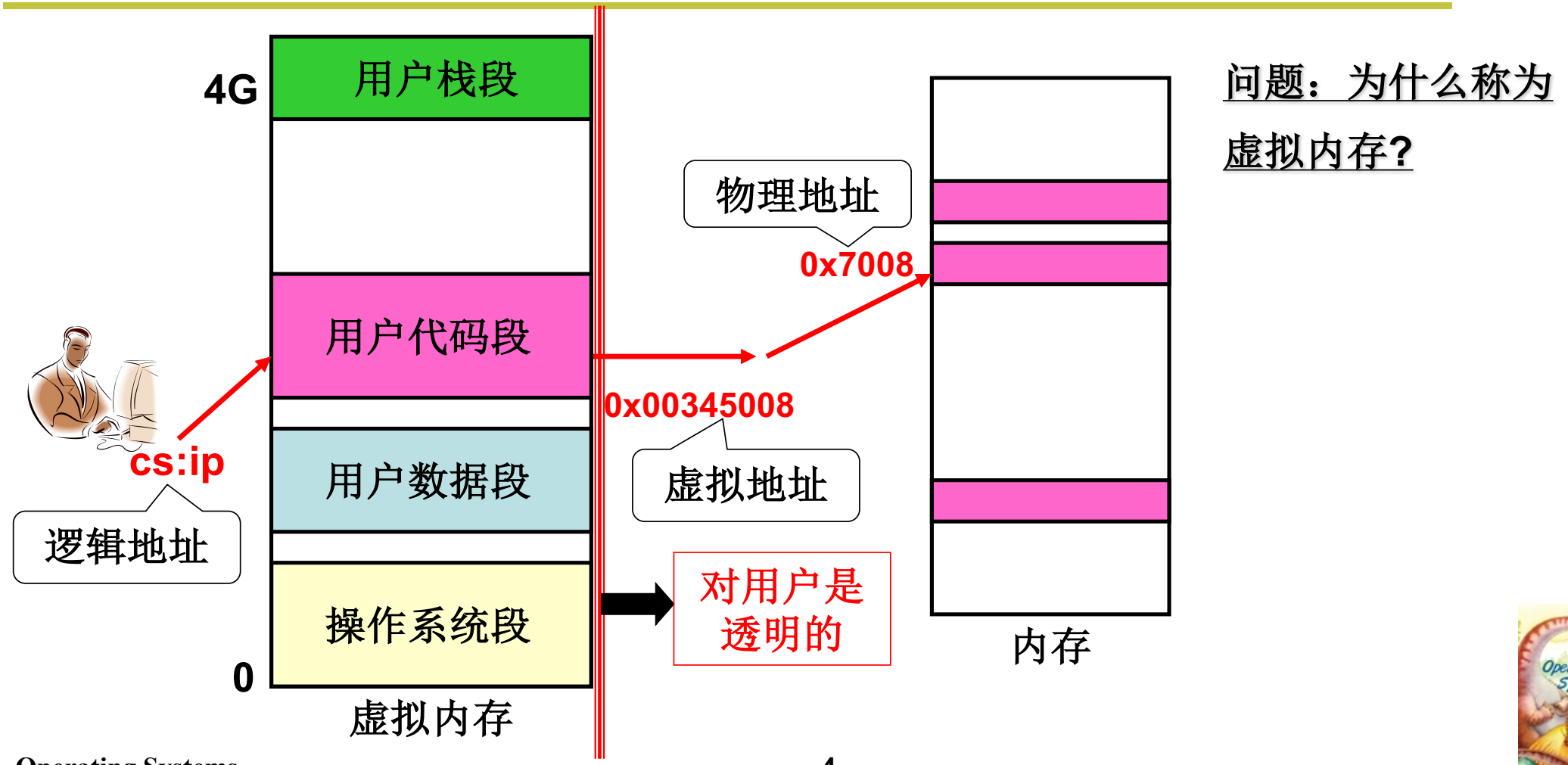
段、页结合: 程序员希望用段，物理内存希望用页，所以...

关键在于虚拟内存这个概念，
作为一个抽象层
将段和页结合起来

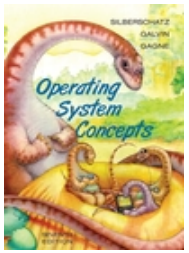




段、页同时存在：段面向用户/页面向硬件



问题：为什么称为虚拟内存？



段、页同时存在是的重定位(地址翻译)

段号+偏移(cs:ip)

逻辑地址

段号	基址	长度	保护
0	0x4000	0x0800	R
1	0x4800	0x1400	R/W
2	0xF000	0x1000	R/W
3	0x0000	0x3000	R

称为虚拟地址

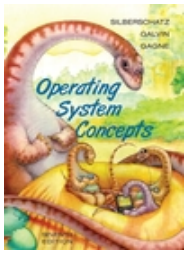
页号 | 偏移

物理地址

物理页号 | 偏移

页框号	保护
5	R
1	R/W
3	R/W
7	R

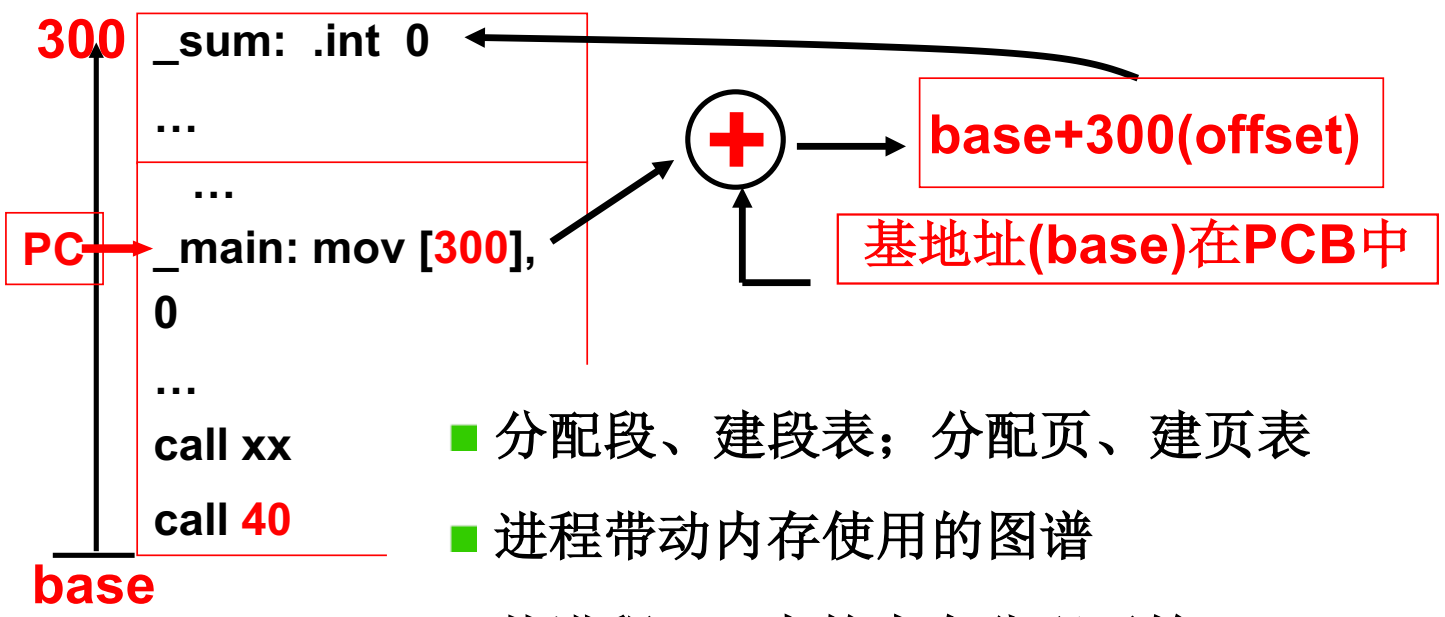
两层地址翻译
第一层是基于段的地址翻译
映射到虚拟地址
第二层是基于页的地址翻译
映射到物理地址

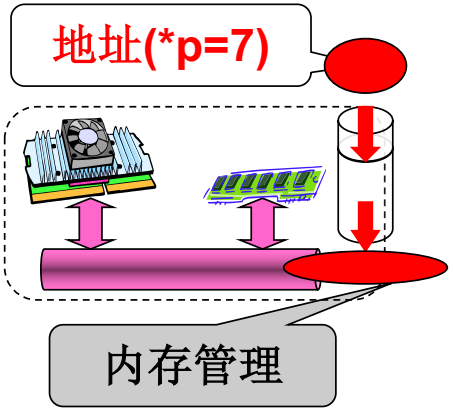


一个实际的段、页式内存管理

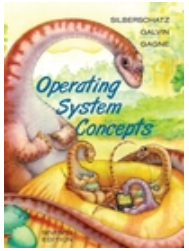
这个故事从哪里开始?

- 内存管理核心就是内存分配，所以从程序放入内存、使用内存开始...

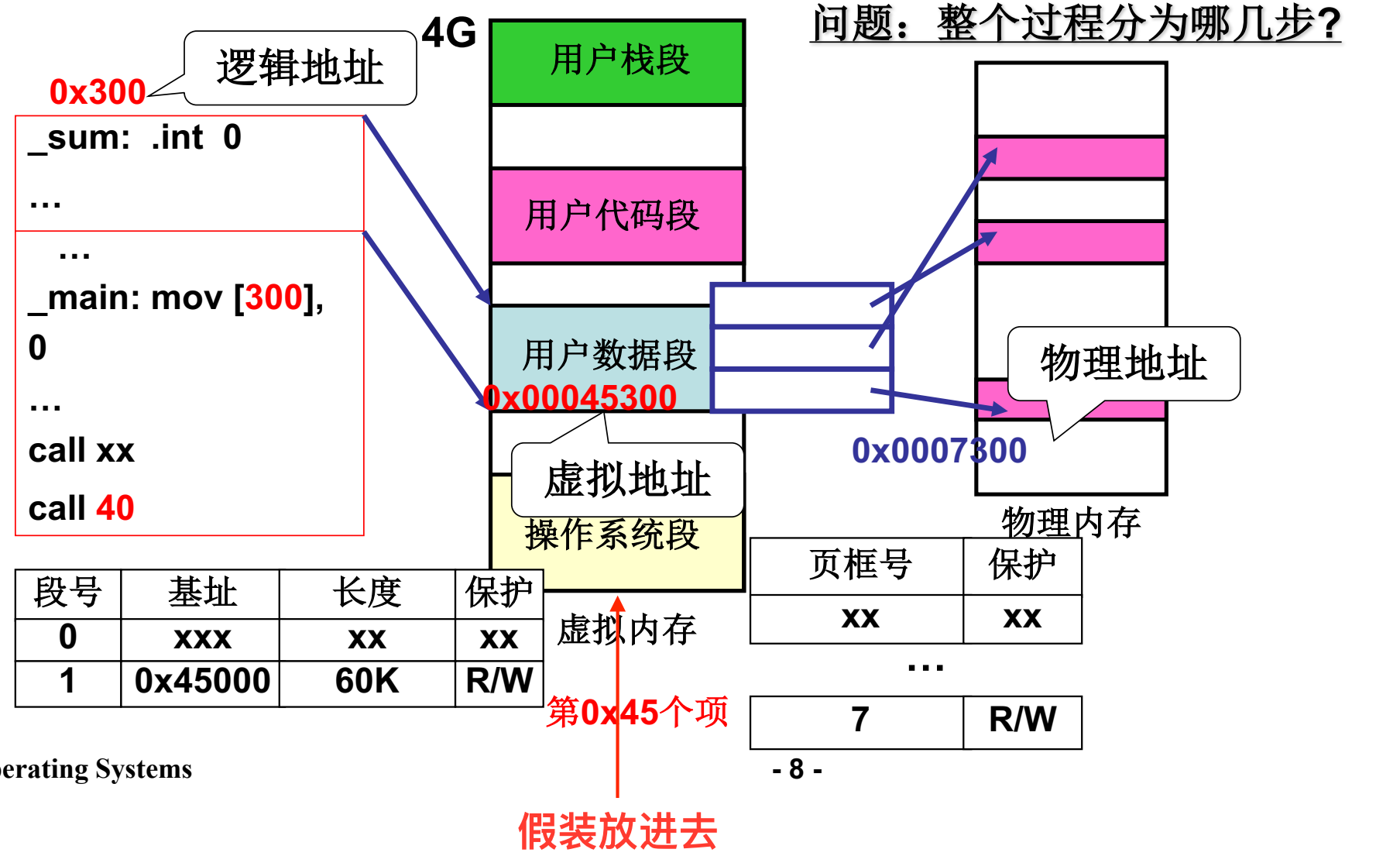


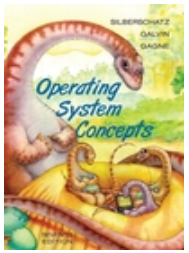


- 分配段、建段表；分配页、建页表
- 进程带动内存使用的图谱
- 从进程fork中的内存分配开始..



段、页式内存下程序如何载入内存?





故事从fork()开始　分配虚存、建段表

- fork()→sys_fork→copy_process的路都已经走过了

```
在linux/kernel/fork.c中
int copy_process(int nr, long ebp,...)
{    ...
     copy_mem(nr, p); ...
```

的确是进程带动内存!

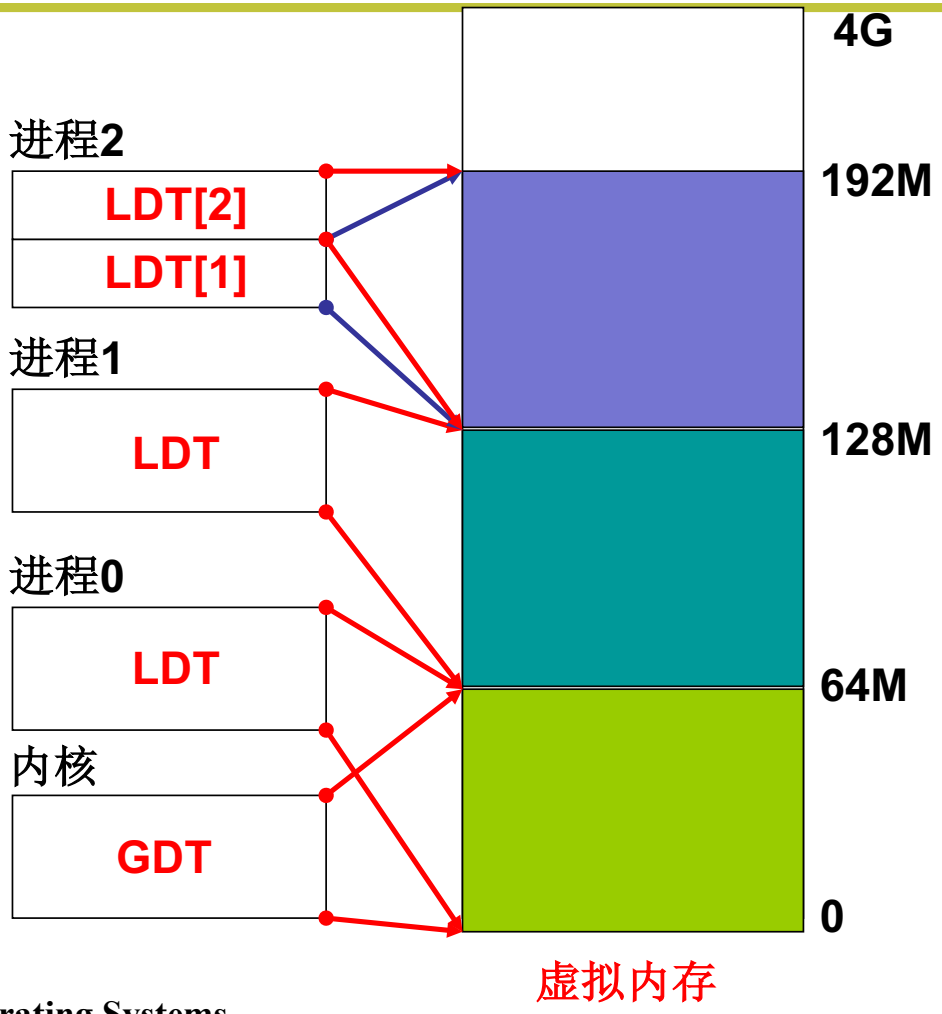
- 现在开始分析当时那个神秘的copy_mem了

```
int copy_mem(int nr, task_struct *p)
{
   unsigned long new_data_base;
   new_data_base=nr*0x4000000;   //64M*nr
   set_base(p->ldt[1],new_data_base);
   set_base(p->ldt[2],new_data_base);
```

PCB

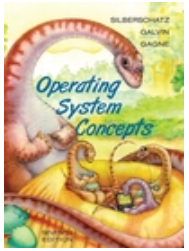
p是什么?进程切换跟着切换

进程0、进程1、进程2的虚拟地址

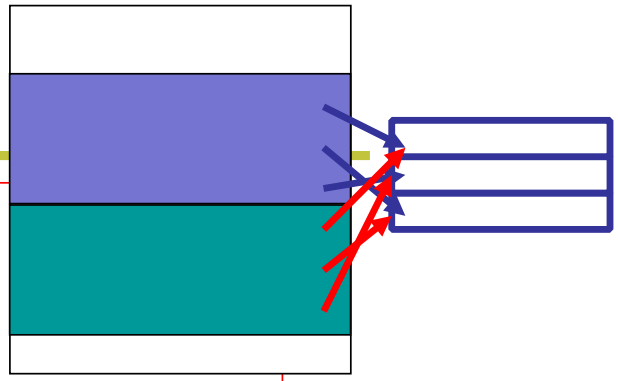


- 每个进程的代码段、数据段都是一个段
- 每个进程占64M虚拟地址空间，互不重叠

问题：这意味着什么样的简化？



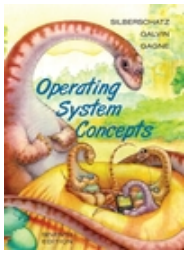
接下来应该是什么了？分配内存、建页表



```
int copy_mem(int nr, task_struct *p)
{   unsigned long old_data_base;
    old_data_base=get_base(current->ldt[2]);
    copy_page_tables(old_data_base,new_data_base,data_limit);
```

```
int copy_page_tables(unsigned long from,unsigned long to, long size)
{   from_dir = (unsigned long *)((from>>20)&0xffc);
    to_dir = (unsigned long *)((to>>20)&0xffc);
    size = (unsigned long)(size+0x3fffff)>>22;
    for(; size-->0; from_dir++, to_dir++){
        from_page_table=(0xfffff000&*from_dir);
        to_page_table=get_free_page();
```

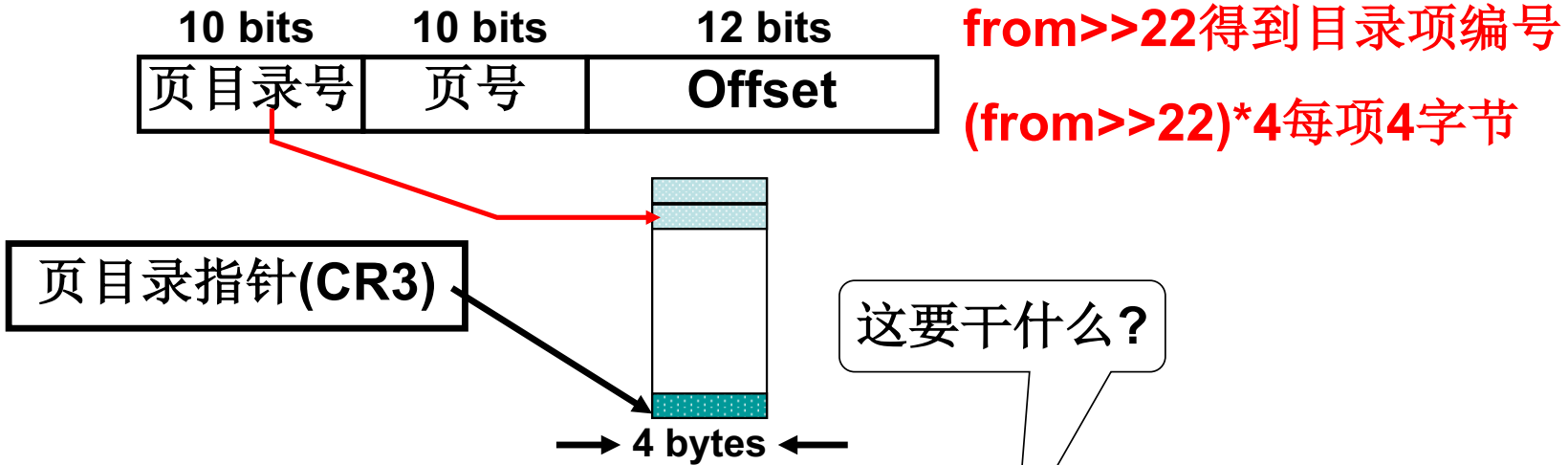
共用父进程的物理页



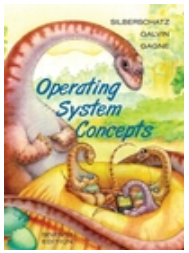
这里的from_dir，to_dir是什么？

```
from_dir = (unsigned long *)((from>>20)&0xffc);
to_dir = (unsigned long *)((to>>20)&0xffc);
size = (unsigned long)(size+0x3fffff)>>22;
```

- from是？ 32位虚拟地址，这个地址的格式是否还记得？

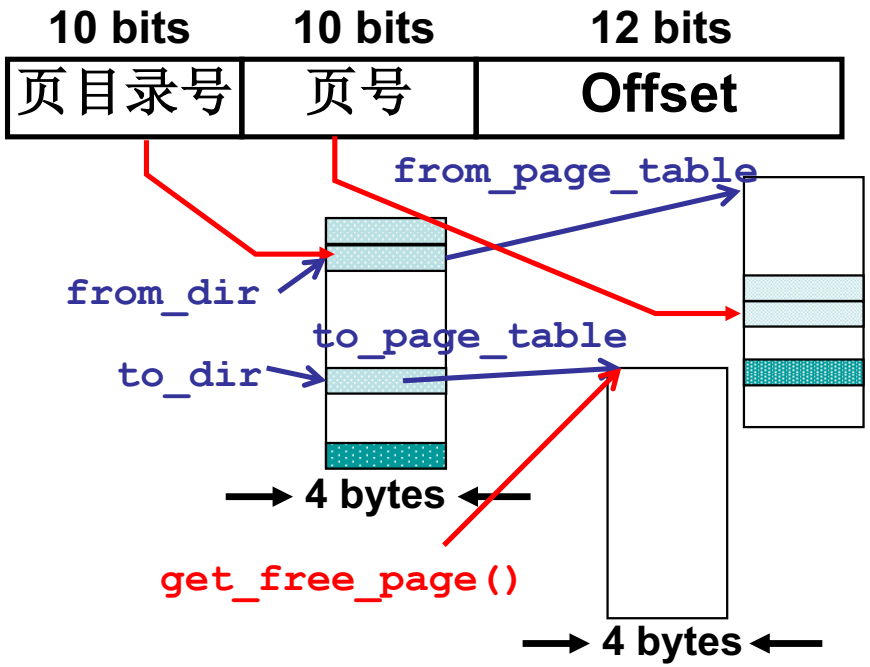



```
for(; size-->0; from_dir++, to_dir++){
        from_page_table=(0xfffff000&*from_dir);
```

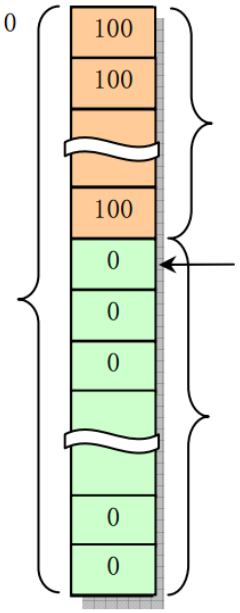


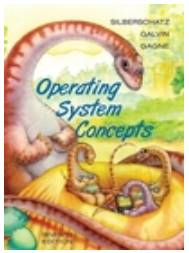
from_page_table与to_page_table?

```
for(; size-->0; from_dir++, to_dir++){
    to_page_table=get_free_page();
    *to_dir=((unsigned long)to_page_table)|7;
```

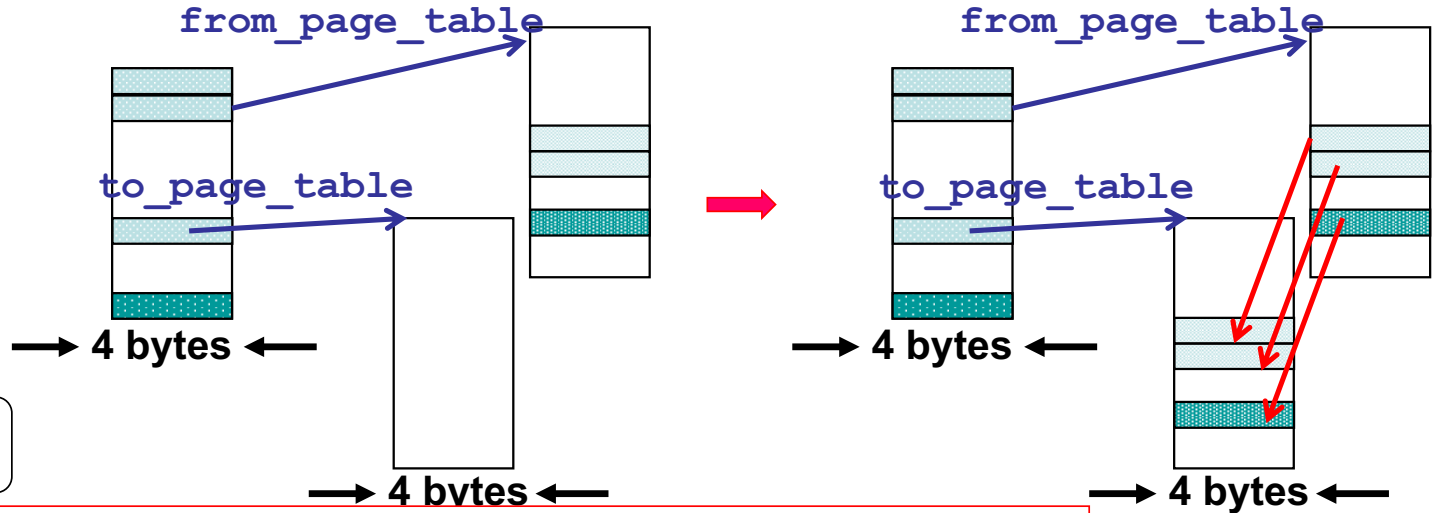



```
unsigned long
get_free_page(voi
{ register unsign
long _res asm("ax
_asm_("std; repne
scasb\n\t"
"movl %%edx,%%eax
"D"(mem_map+PAGIG
GES-1));
return _res; }
```





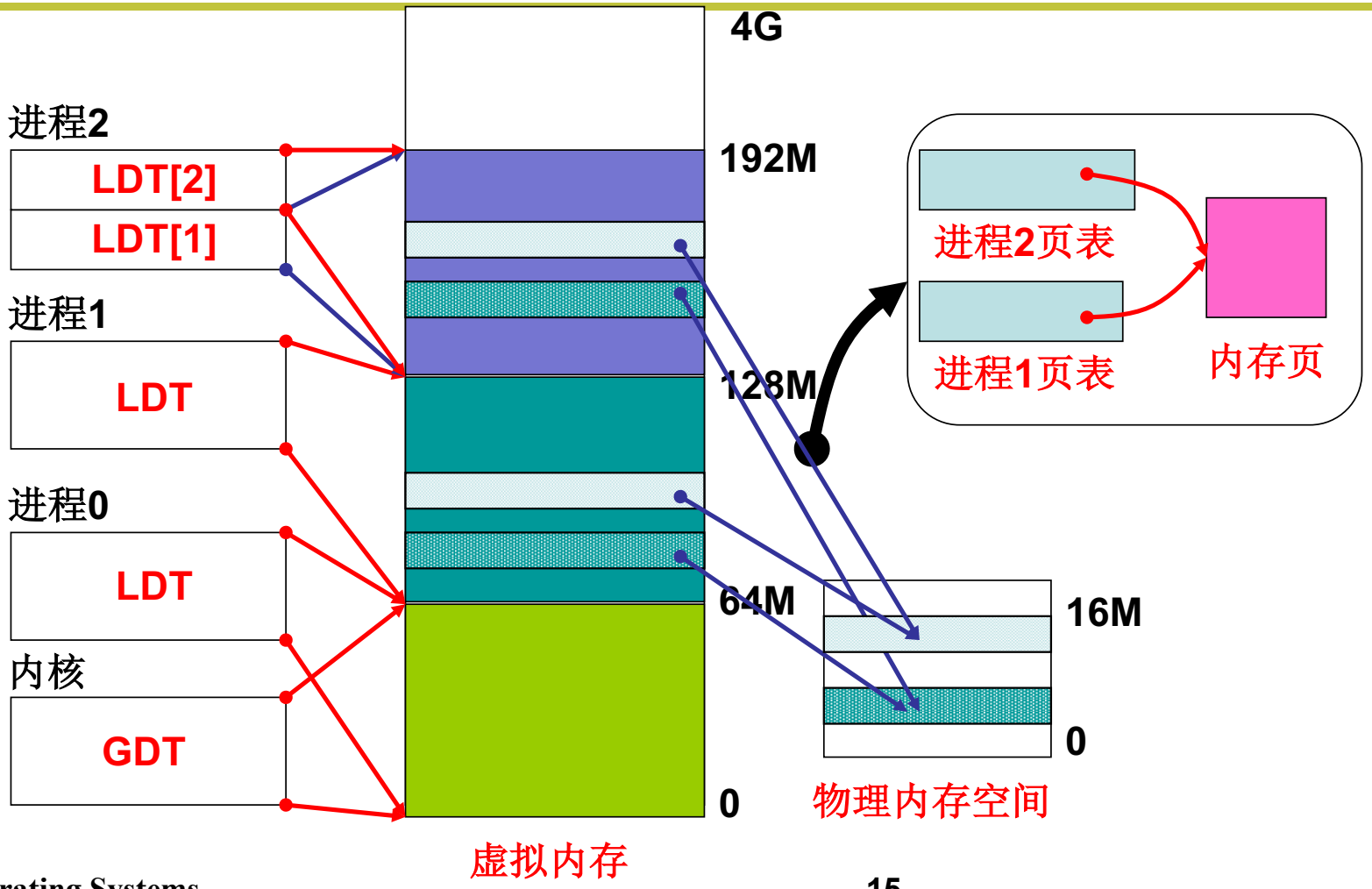
接下来干什么应该猜也猜的到...

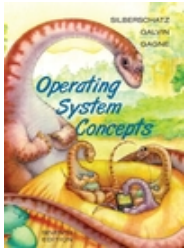


nr=1024!

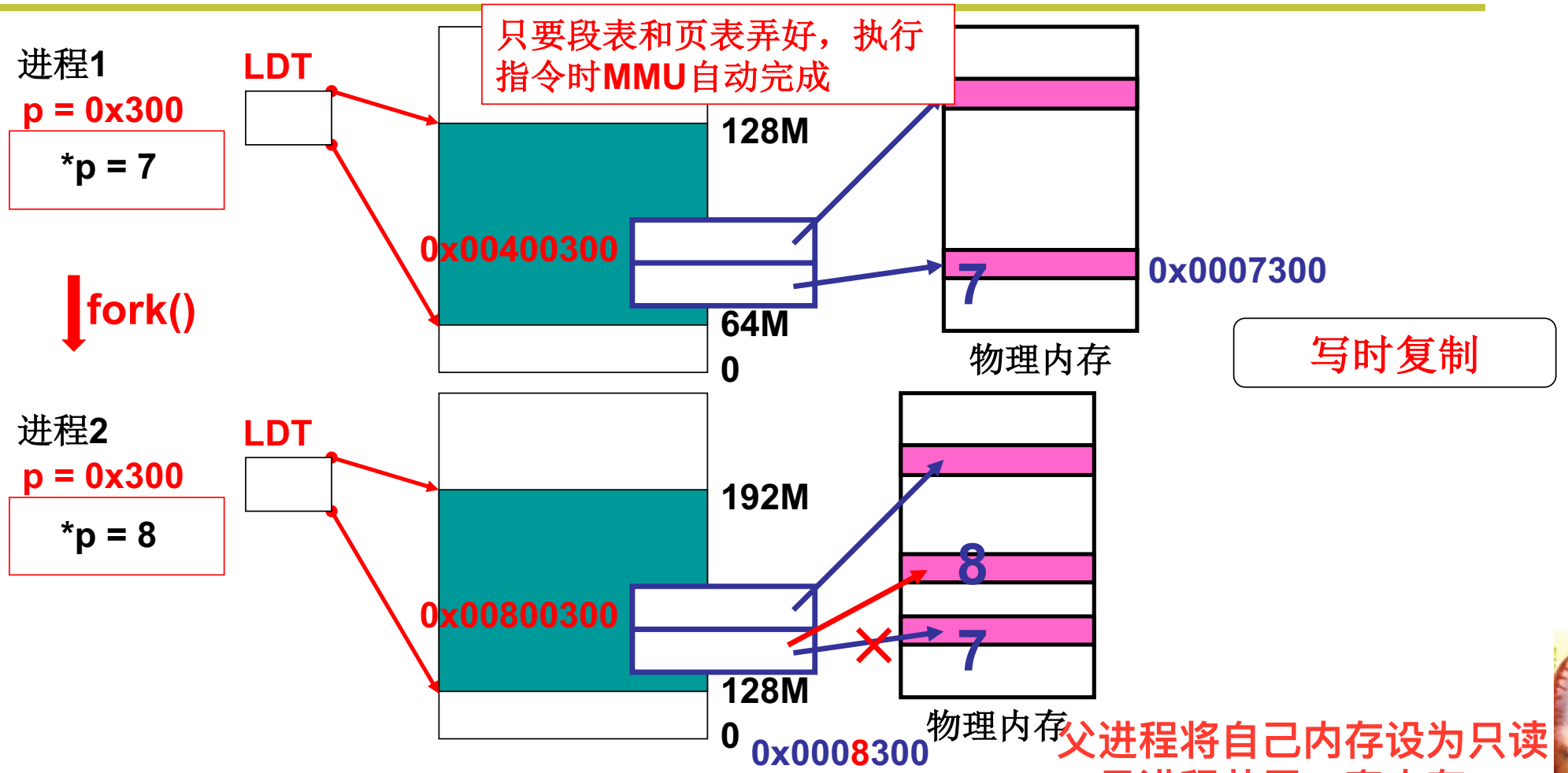
```
for(;nr-->0;from_page_table++,to_page_table++){
  this_page = *from_page_table;
  this_page&=~2;//只读
  *to_page_table=this_page;
  *from_page_table=this_page;
  this_page -= LOW_MEM; this_page >>= 12;
  mem_map[this_page]++; }
```

程序、虚拟内存+物理内存的样子





*p=7? 父进程*p=7、子进程*p=8? 读写内存 *p=7



父进程将自己内存设为只读
子进程共用一套内存，
如果要写的话，则将对应的页复制一份，
这样就实现了子进程的自己的物理
内存区域