**内容:**

**第一章 硬件：计算机组成**

**第二章 静态程序：**

**二进制可执行文件<-(链接器)--目标文件<-(汇编器) --汇编语言<-(编译器)--高级语言**

**第三章 动态程序：进程**

**<-(加载器)—静态程序**

**第四章 虚拟内存**

**第五章 文件系统**

**第六章 并发（多线程和锁）**

**第0章 :引言**

**1,写出代码实现如下函数：**

**/\* Return 1 when x contains an even number of 1s; 0 otherwise.**

**Assume w=32 \*/**

**int even\_ones (unsigned x);**

**你的代码最多只能包括12个算术运算、位运算和逻辑运算。**

**C语言中的位运算：&, |, ~, ^ (与、或、非、异或)；移位运算<<，>>**

答案: int even\_ones (unsigned x) {

x=x&(x>>16);

x=x&(x>>8);

x=x&(x>>4);

x=x&(x>>2);

x=x&(x>>1);

return !(x&1);

}

**第一章：硬件：计算机组成**

**1，硬件：中央处理单元（CPU），存储单元（内存），输入输出设备（外存、网卡），总线2，存储器类型：RAM：random access memory可读可写 掉电内容丢失**

**ROM：read-only memory只读 永久保存内容**

**3，CPU的3大主要部件：,运算器，控制器，寄存器组 (Cache)**

**3.1通用寄存器：EAX、EBX、ECX、EDX都是用来保存数据的, 但各有用途**

**AX : 累加，BX : 基址，CX : 计数，DX : 数据**

**3.2段寄存器和逻辑段**

**CS：IP=CS\*16+IP，当前指令地址**

**DS：[x]=DS\*16+x，当前数据地址**

**SS：SP=SS\*16+SP，当前栈顶地址**

**需自己保证push和pop不越界**

**80x86不允许直接将立即数送入段寄存器**

**Mov ds, 1000H非法**

**Mov ax, 1000H mov ds, ax合法**

**3.3** **8086的系统启动**

**重置系统状态，执行的第一条指令位于FFFF0H**

**CS = FFFFh**

**IP = 0000h**

**其余各个寄存器清零**

**执行ROM中指令，进入BIOS程序的处理**

**系统内存检查**

**初始化中断向量表和相关数据**

**从磁盘中载入操作系统**

**位于磁盘的启动扇区**

**DOS**

**载入执行command.com**

**4，输入/输出设备：键盘、显示器、磁盘**

**I/O接口电路：外设与CPU的桥梁**

**电路插板（卡）：与外设接口插座，与系统I/O总线插槽**

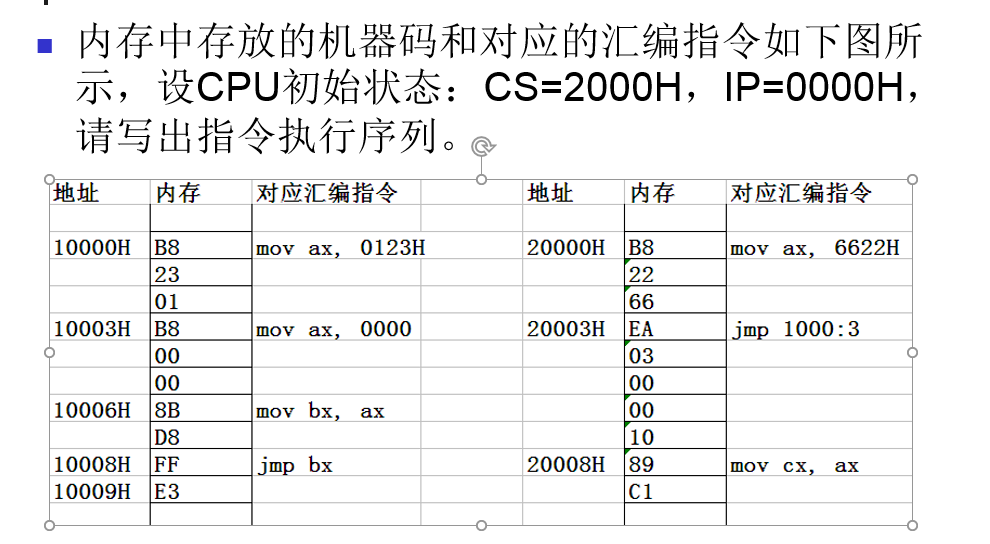
**一：考题：**

**地址：4000H, 4001H, 4002H, 4003H**

**数据：B3H, 79H, 5AH, 93H**

**大端方式，word ptr [4000H]的真实值？**

**8086下，要访问54000H这一单元，段地址的取值范围？**

**二：**

**答案：Mov ax, 6622H**

**Jmp 1000:3**

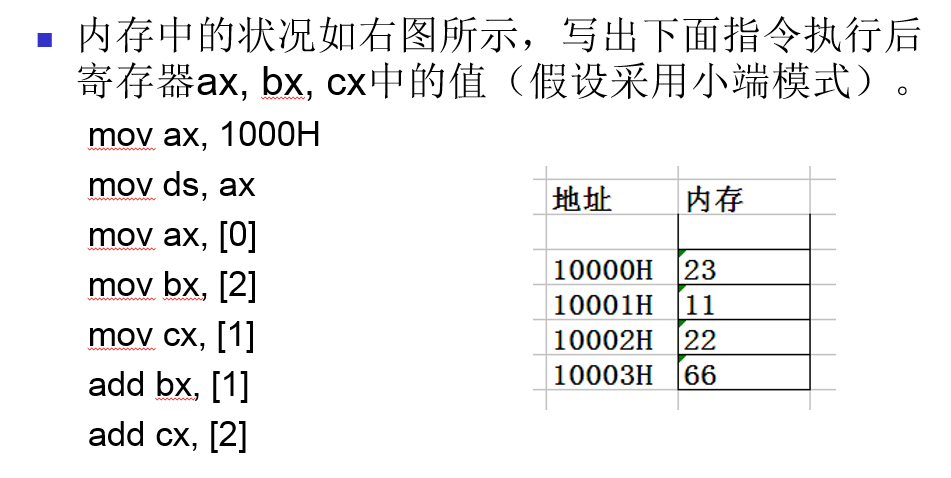
**Mov ax, 0000**

**Mov bx, ax**

**Jmp bx**

**Mov ax, 0123H**

**第3步**

**三：**

**答案：Ax=1000H**

**Ds=1000H**

**Ax=1123H**

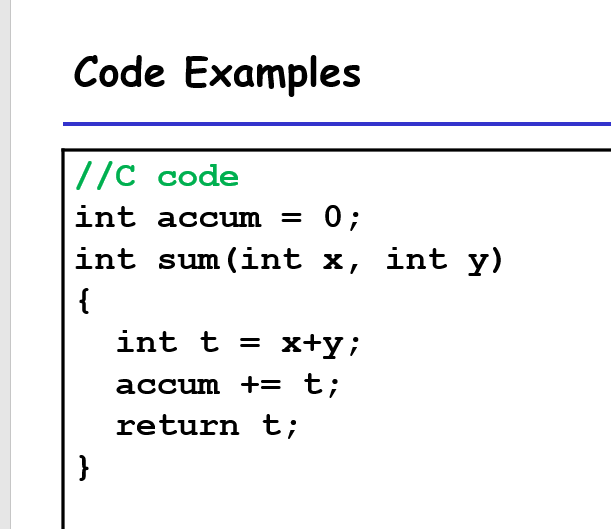
**Bx=6622H**

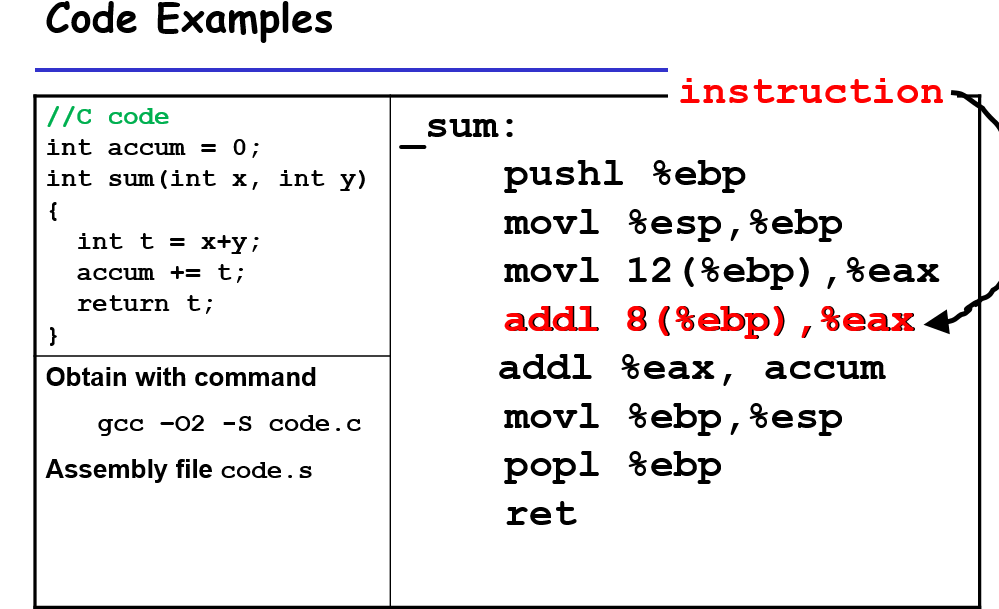
**Cx=2211H**

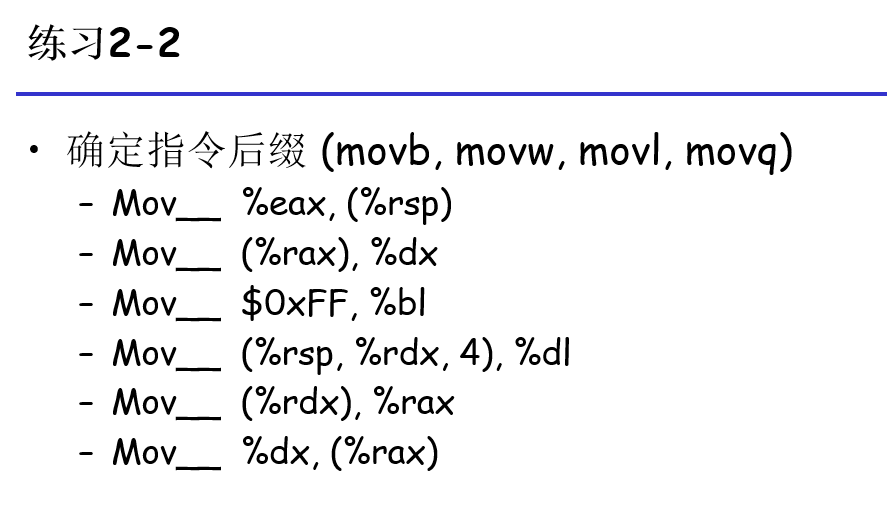
**Bx=8833H**

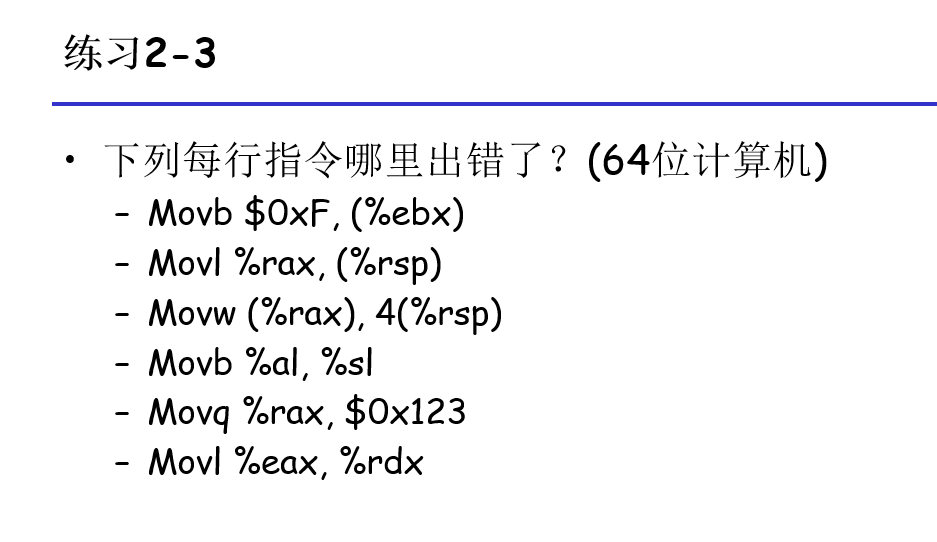
**Cx=8833H**

**第二章：程序与汇编语言**





**二：**

**三：**

**答案：下列每行指令哪里出错了？(64位计算机)**

**Movb $0xF, (%ebx) - %ebx装不下地址**

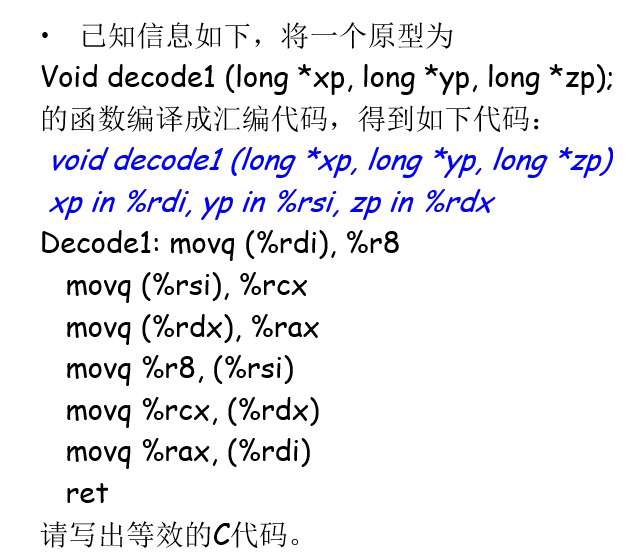
**Movl %rax, (%rsp) – 长度不匹配**

**Movw (%rax), 4(%rsp) – 两个内存**

**Movb %al, %sl – 没有%sl**

**Movq %rax, $0x123 – 不能写入立即数**

**Movl %eax, %rdx – 寄存长度不匹配**

**四：**

**答案：void decode1 (long \*xp, long \*yp, long \*zp) {**

**long x = \*xp;**

**long y = \*yp;**

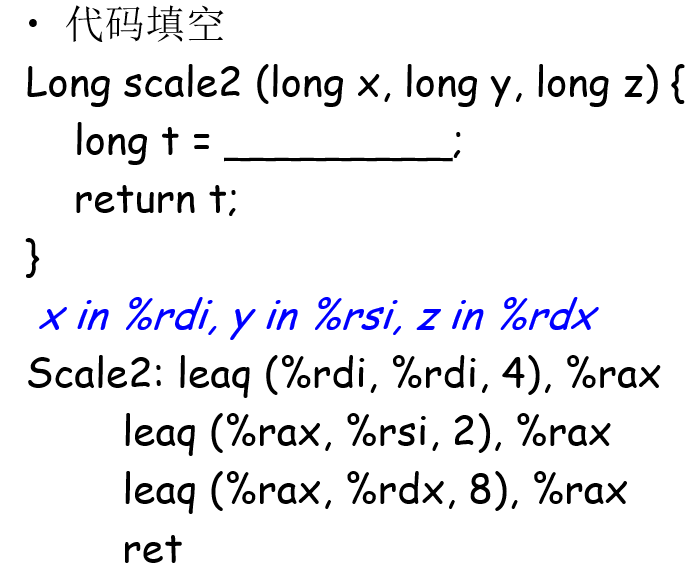
**long z = \*zp;**

**\*yp = x;**

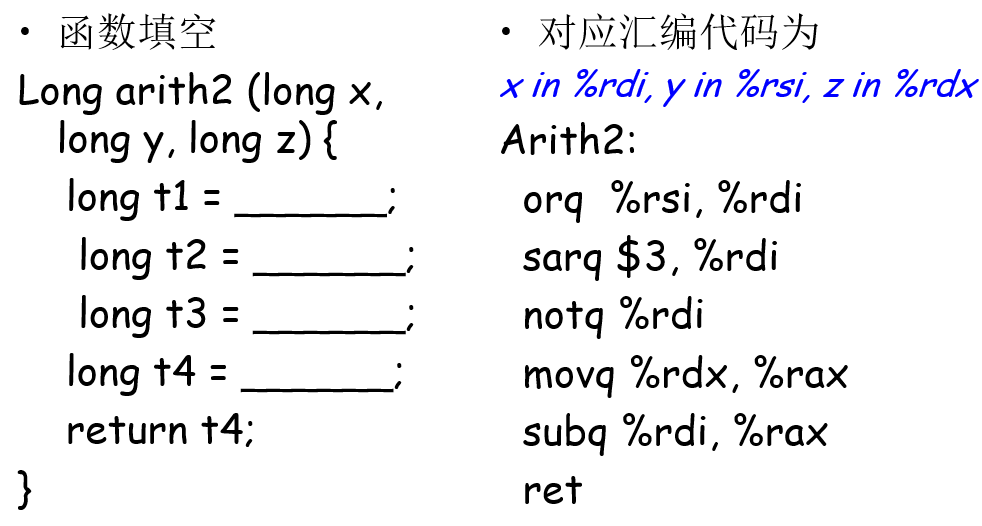
**\*zp = y;**

**\*xp = z;**

**}**

**五：**

**答案：Long t = 5\*x + 2\*y + 8\*z**

**六：**

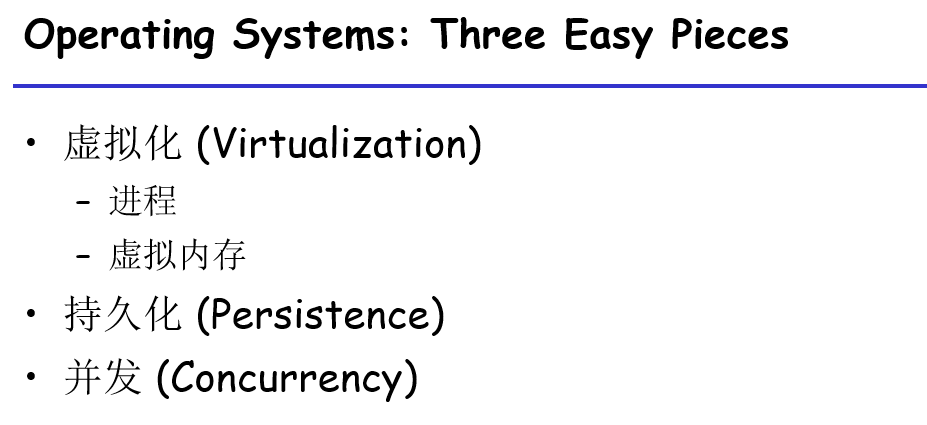
**答案：Long t1 = x ! Y;**

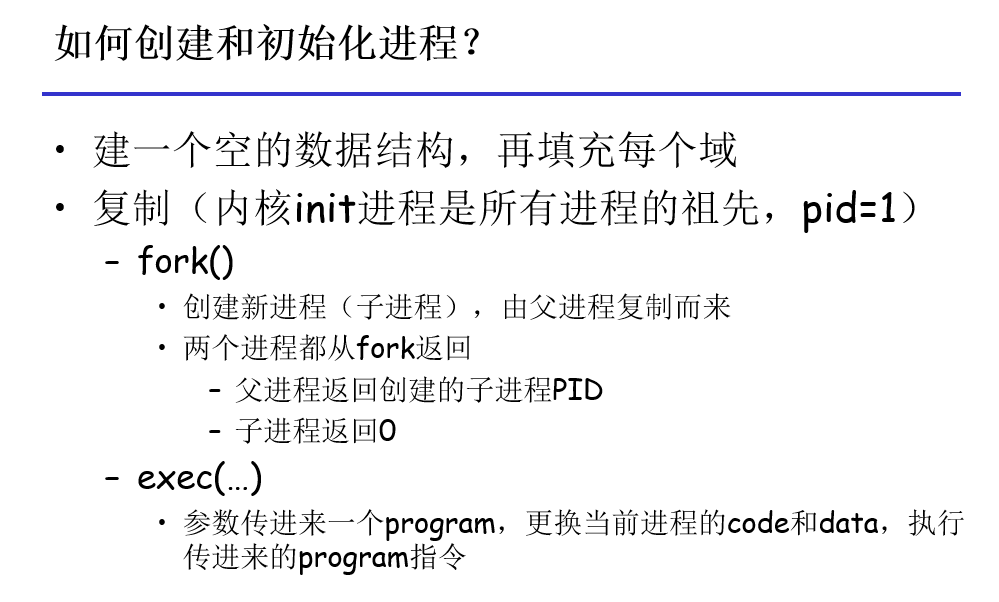
**Long t2 = t1>>3;**

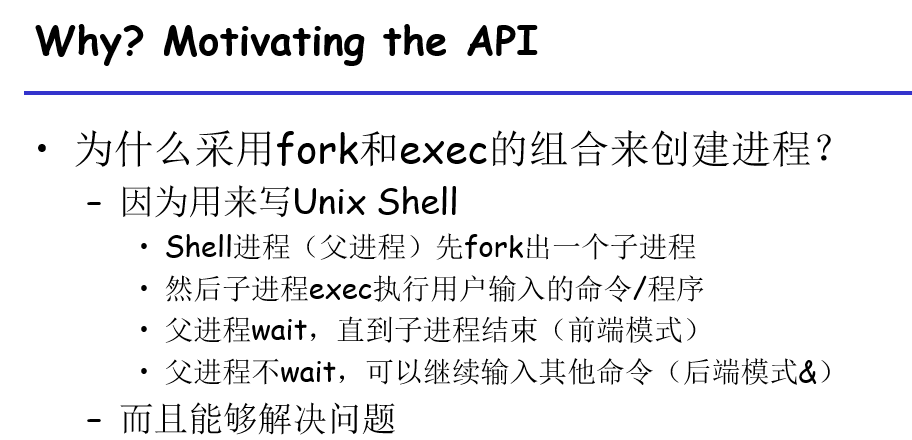
**Long t3 = ~t2;**

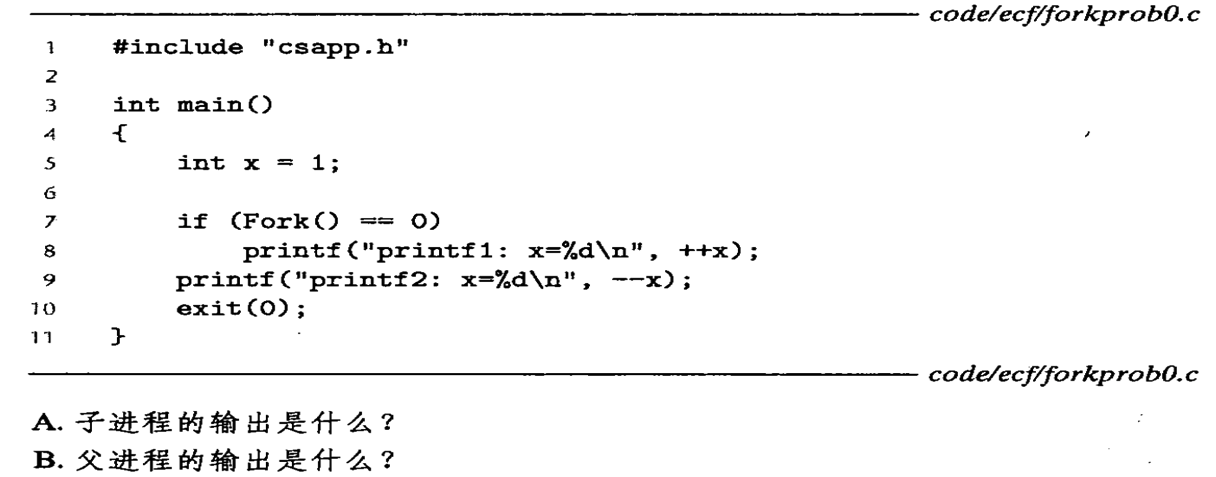
**Long t4 = z-t3;**

**第三章：进程**







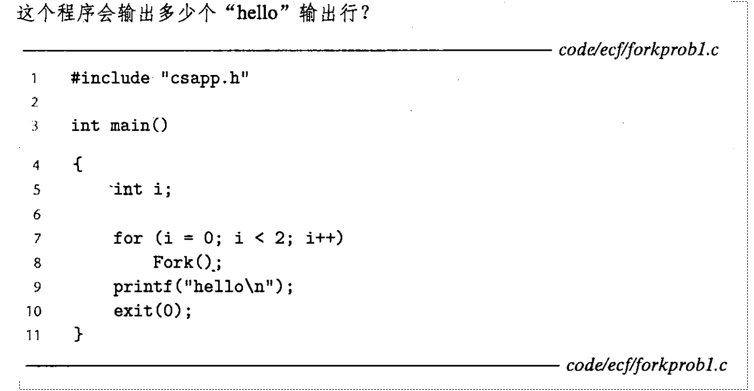
**一：**

**答案：**

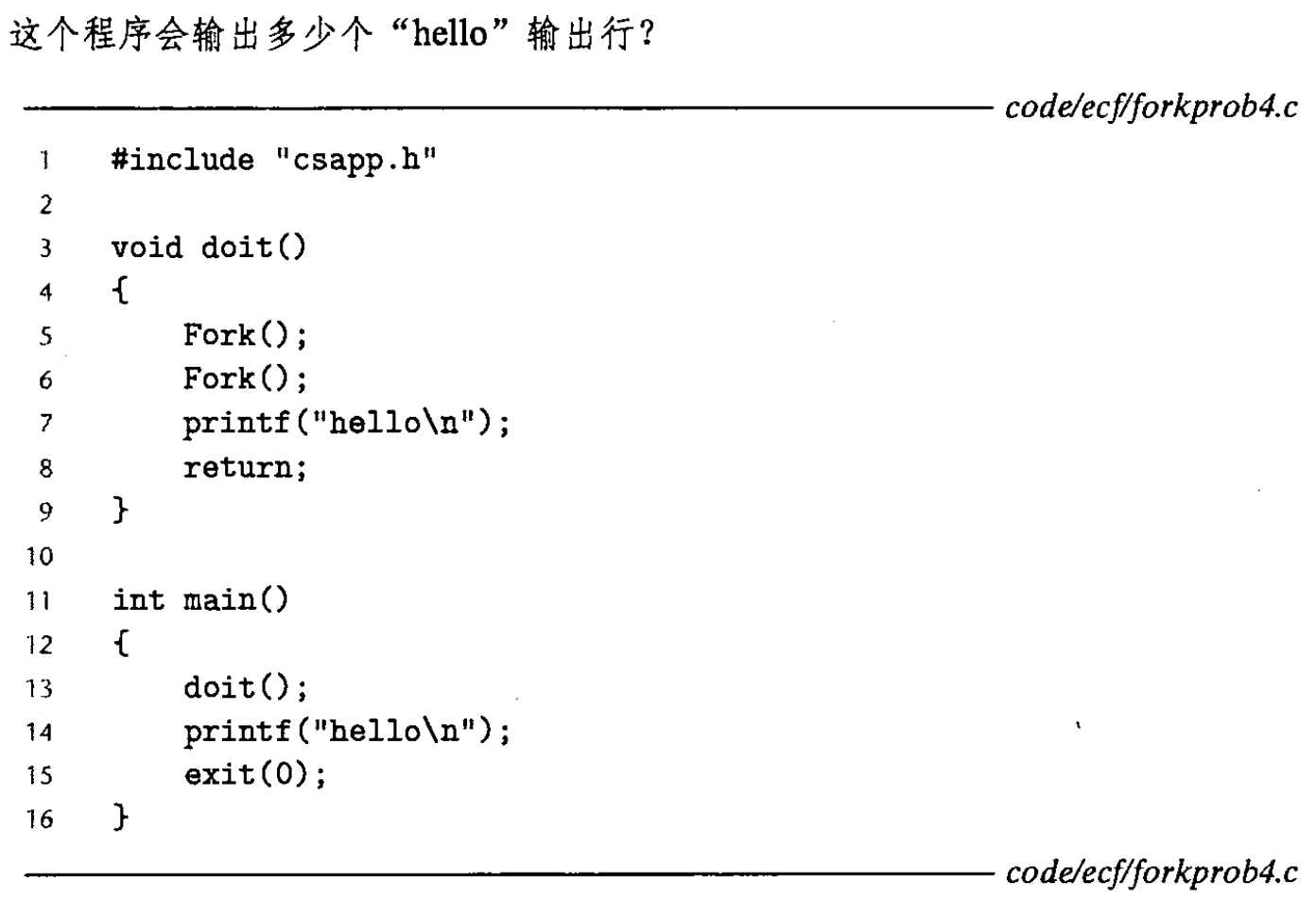
**(a++和++a的区别，前者本身的值是a，而后者的值是a+1)**

**子进程：x=2, x=1**

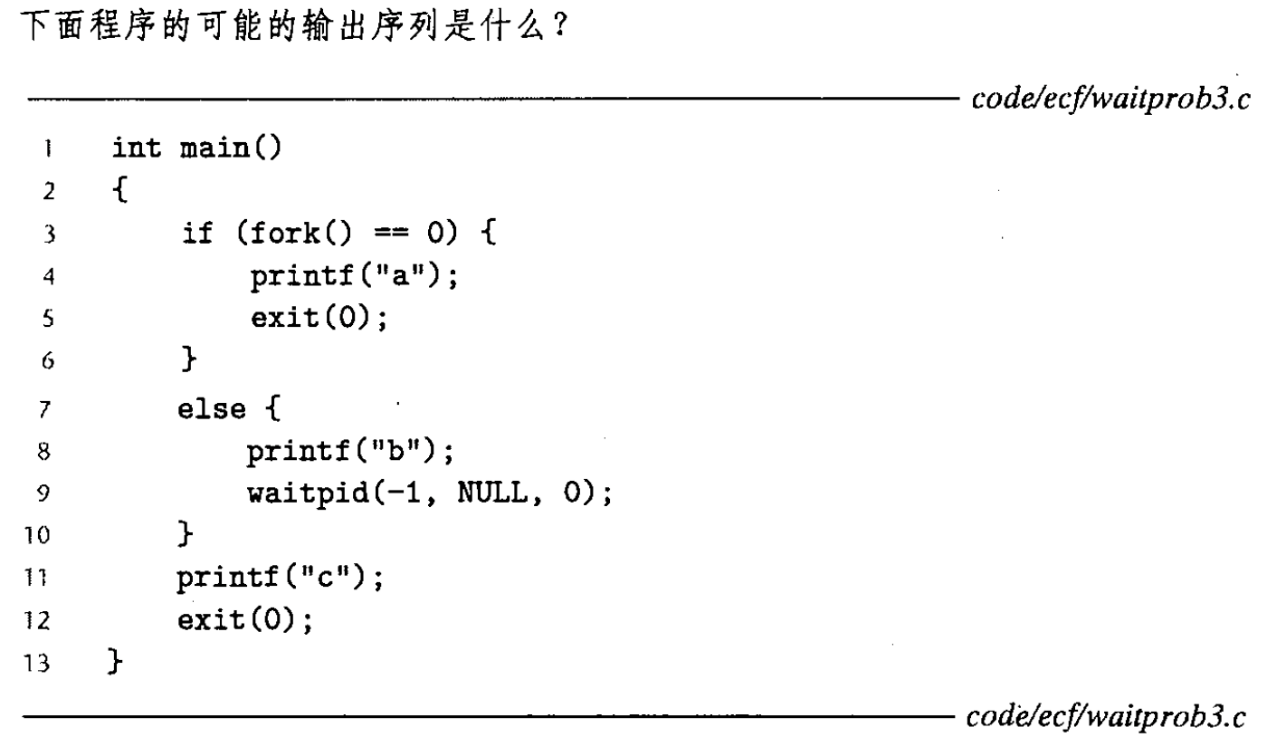
**父进程：x=0**

**二：**

**答案：4个**

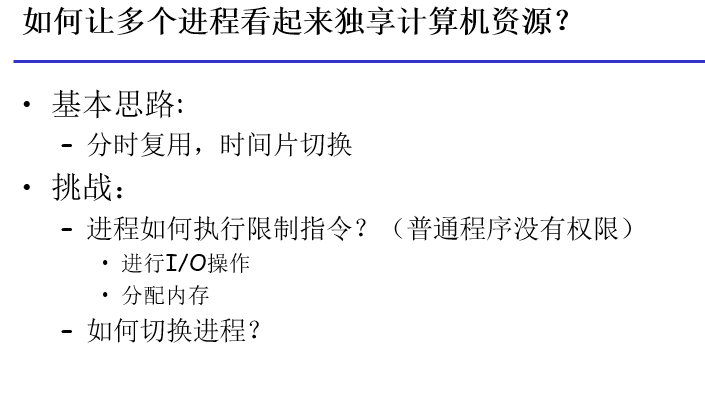
**三：**

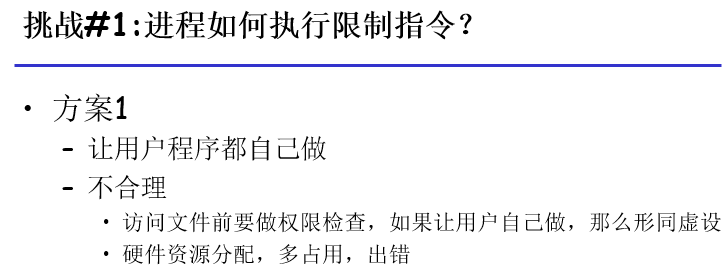
**答案8个 。共4个进程，每个进程在doit()和main()中各输出一个hello**

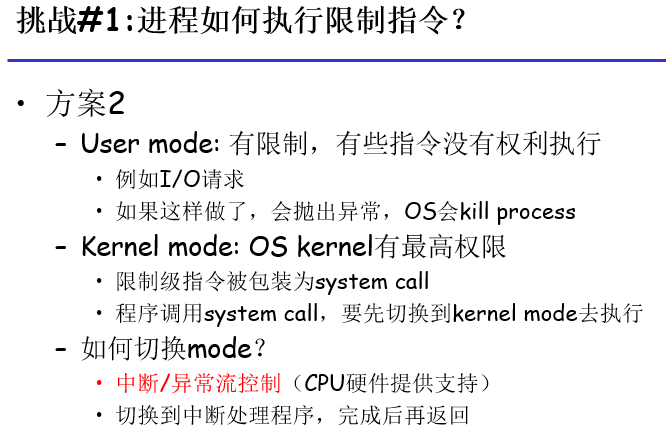
**四：**

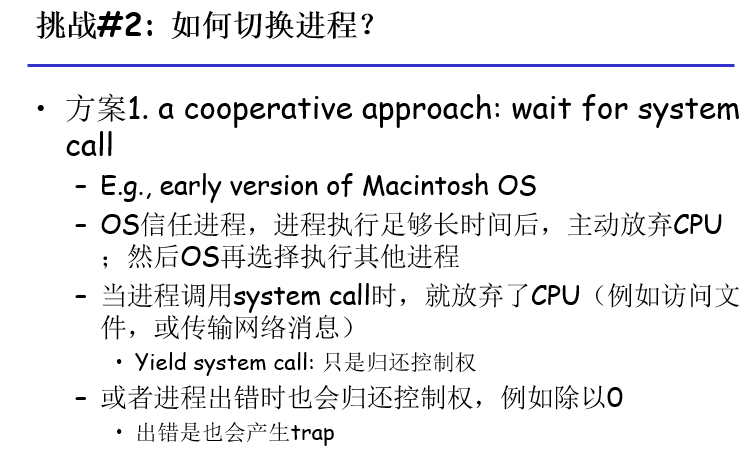
**答案：{A；B}->C的组合（waitpid保证父进程不能在子进程结束前输出c）**

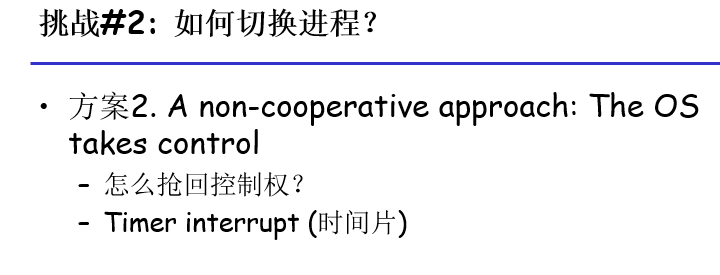
**Abc, bac**

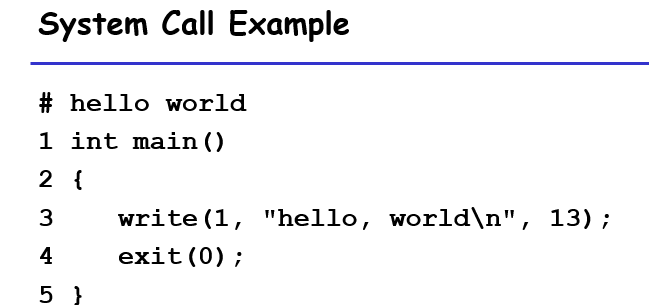


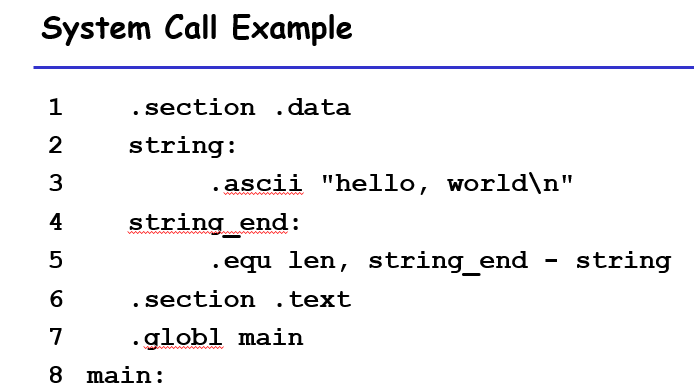


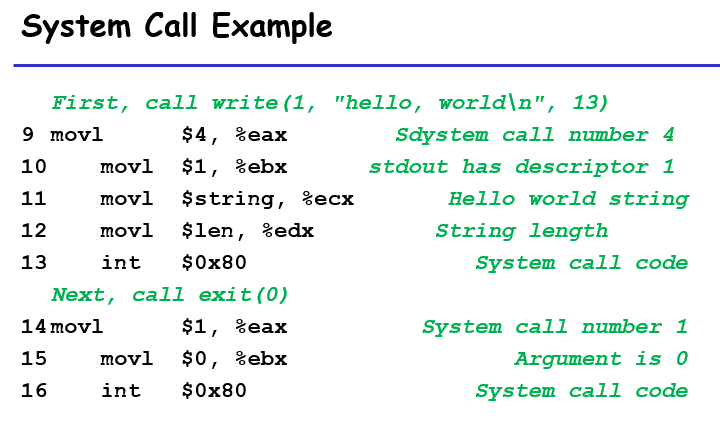


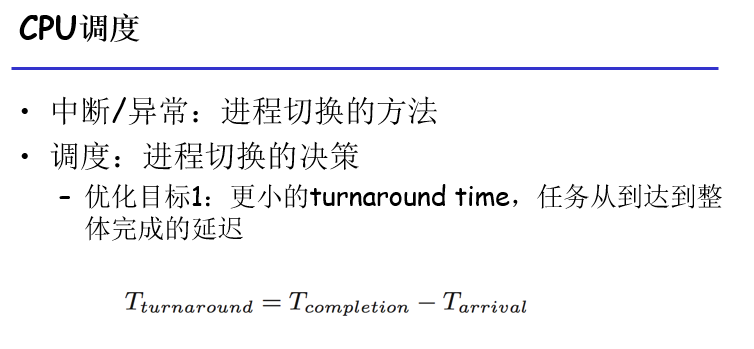


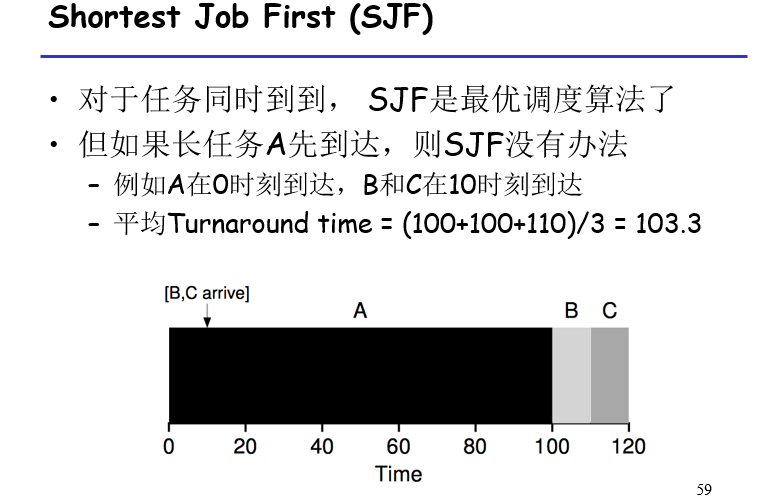


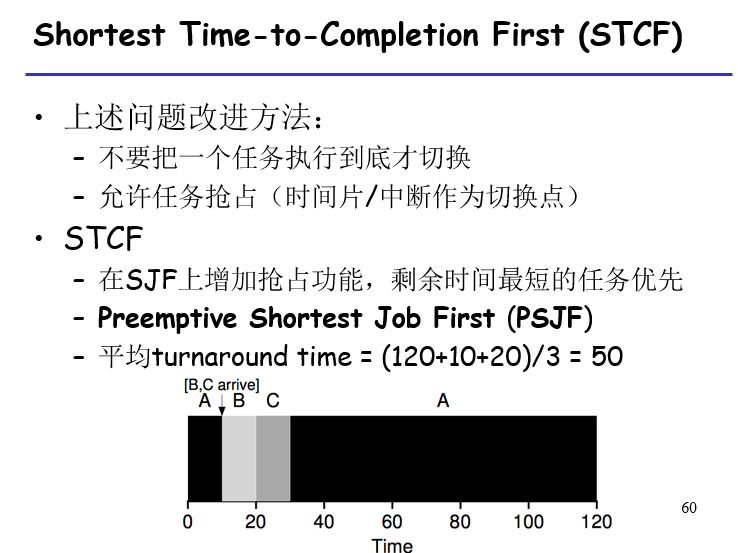


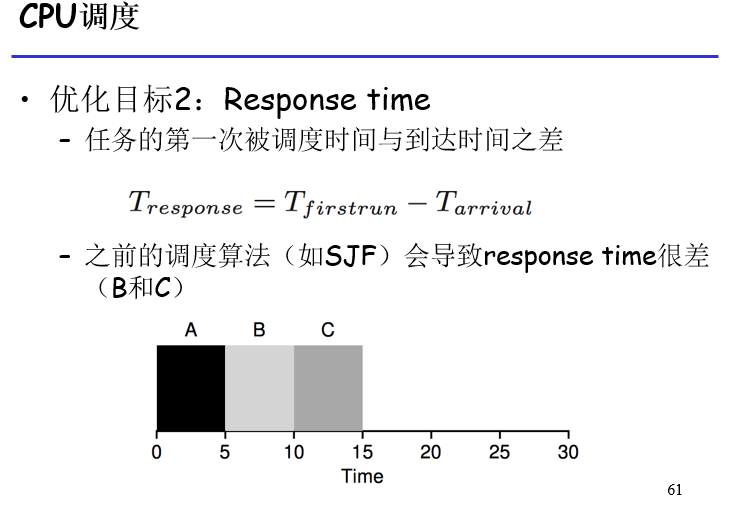


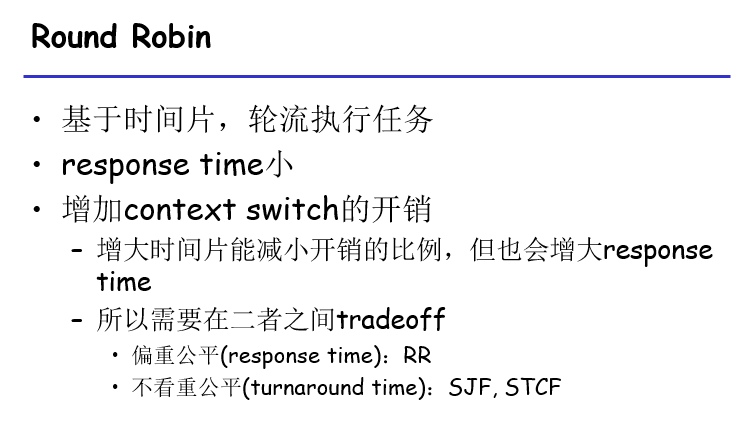


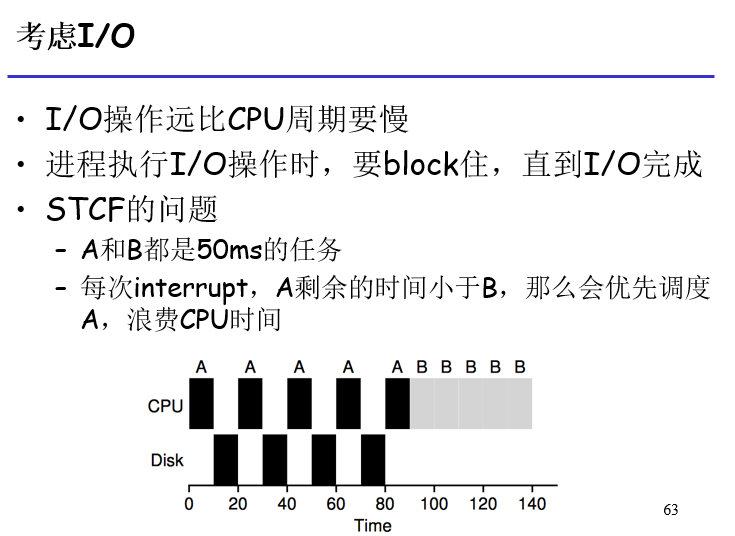


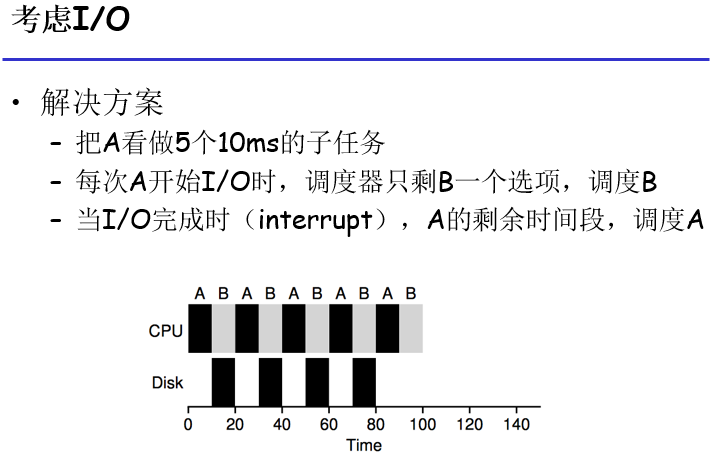


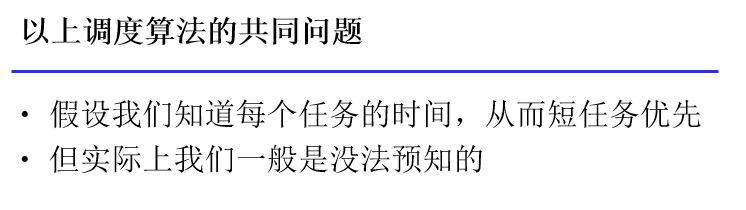


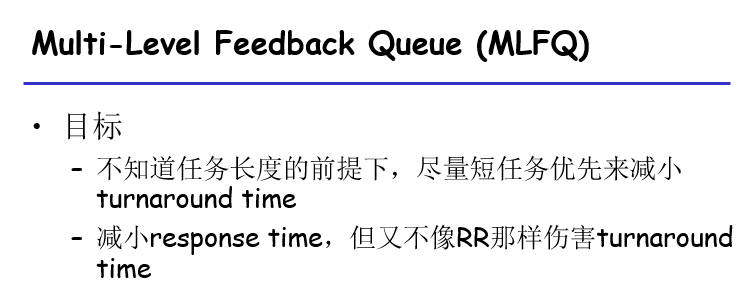


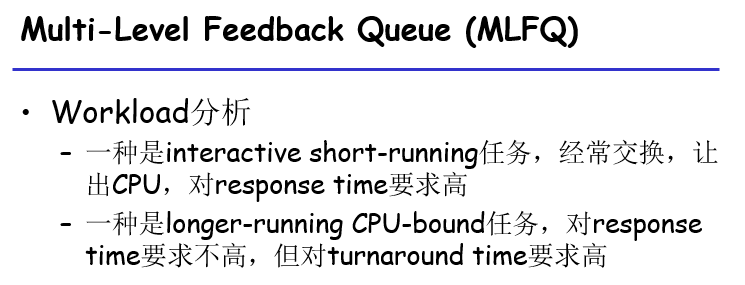


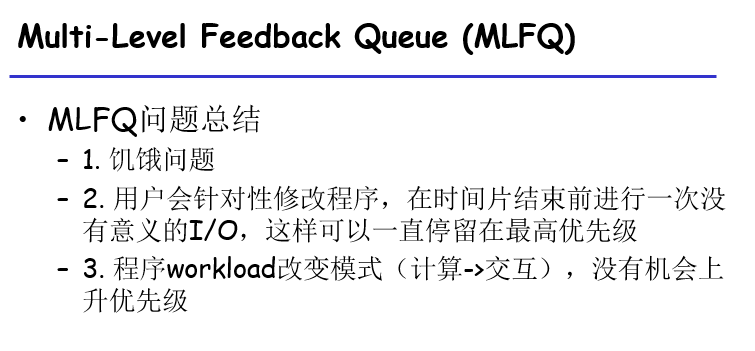


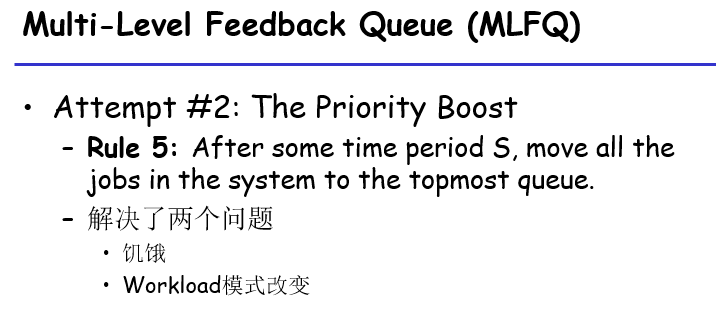




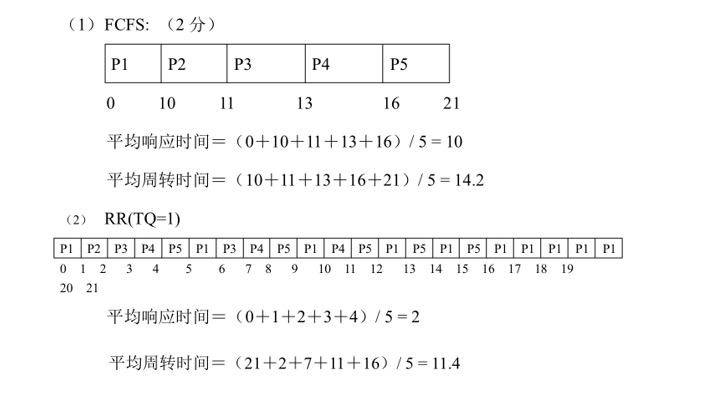


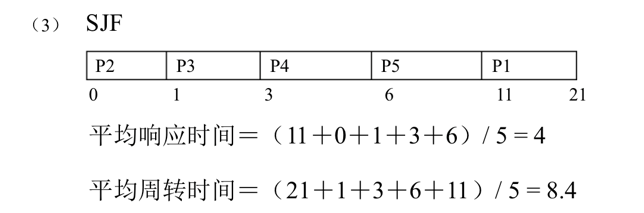




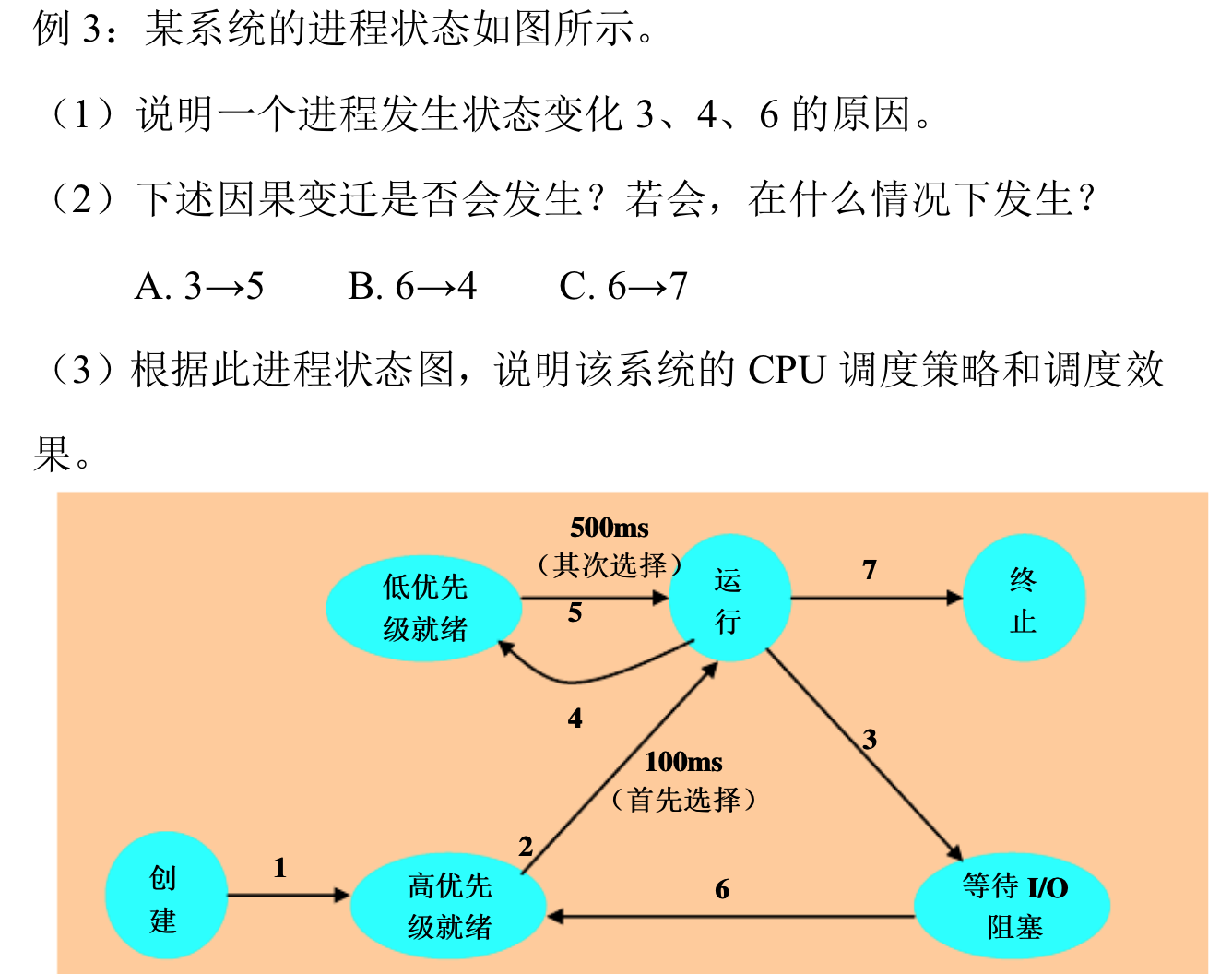


**五：**

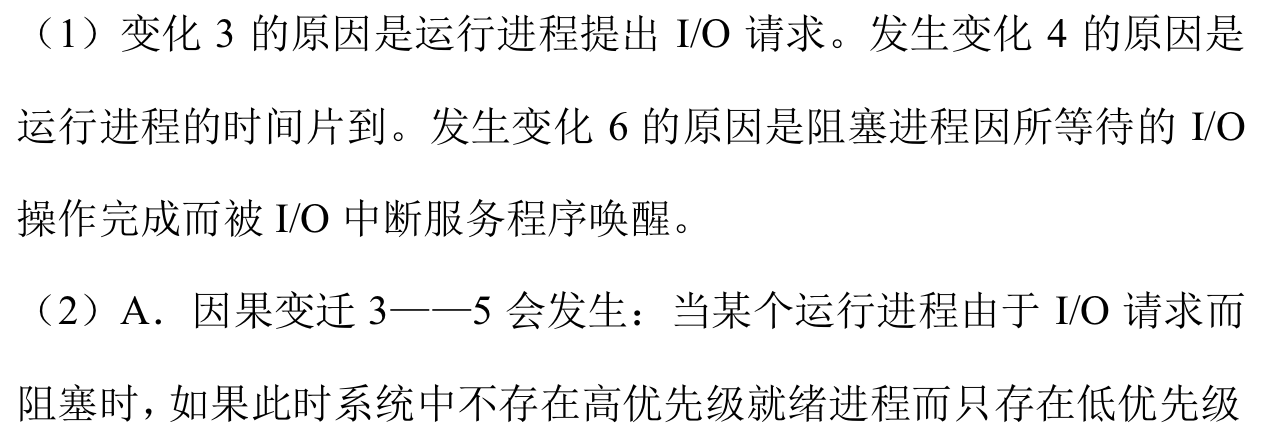


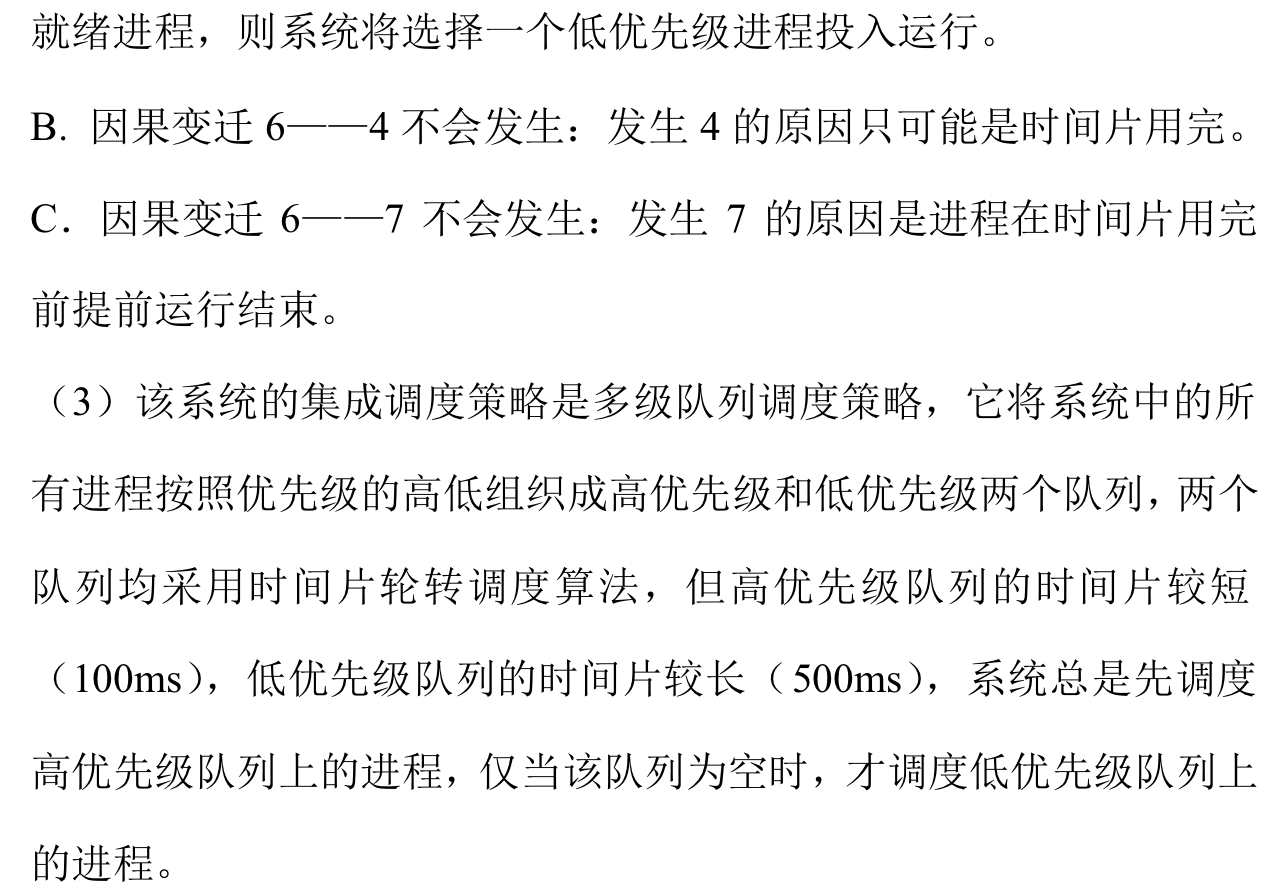


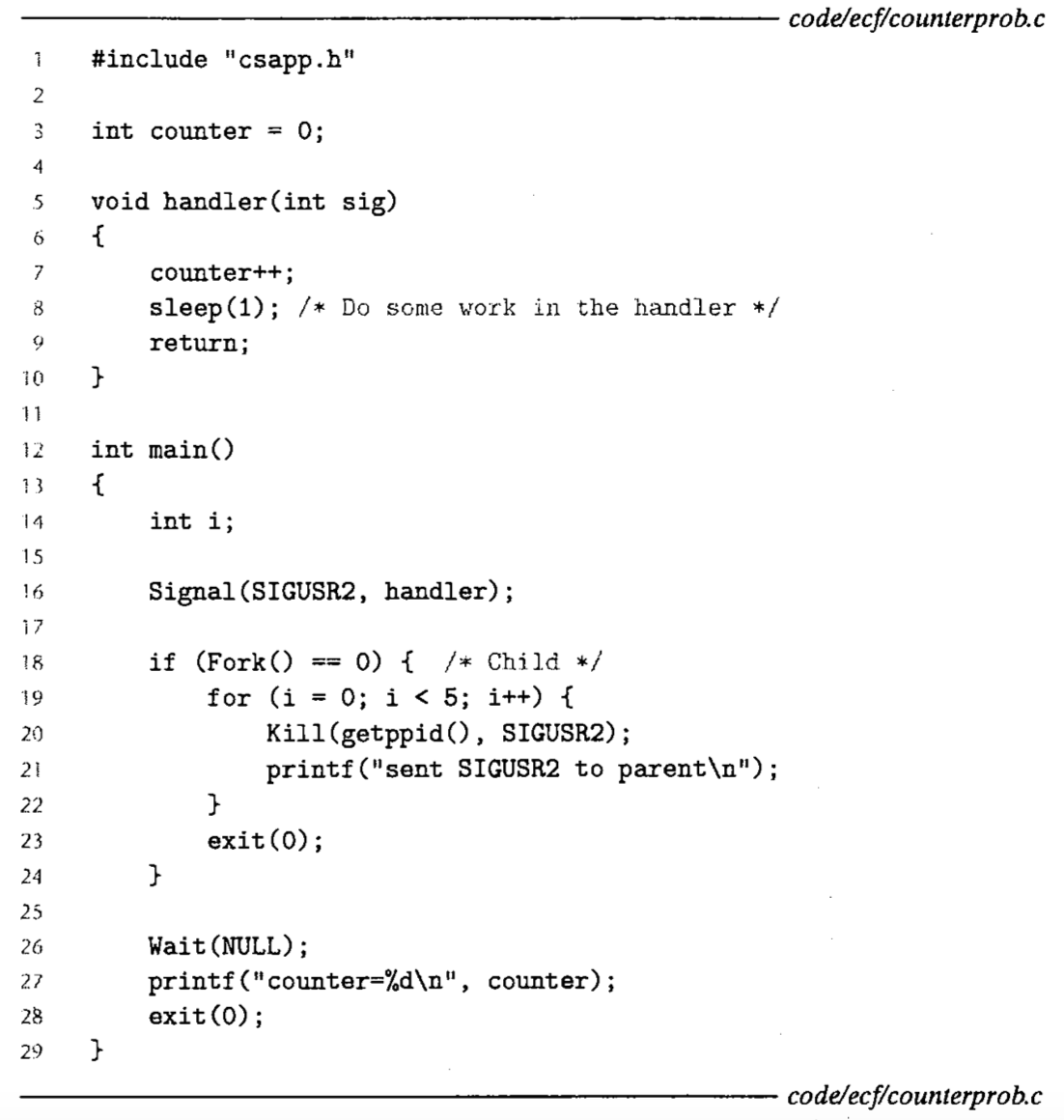
**六：**

****

**答案：**

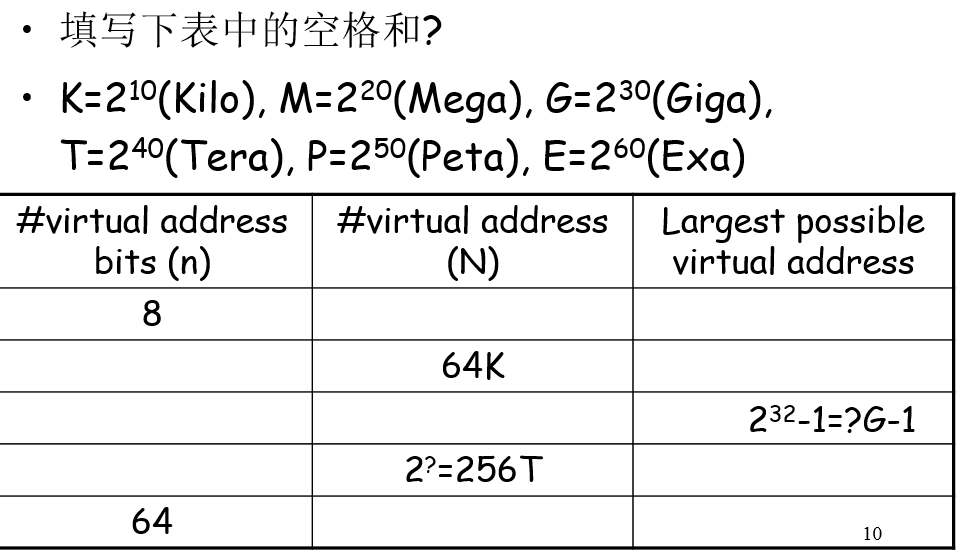
****

****

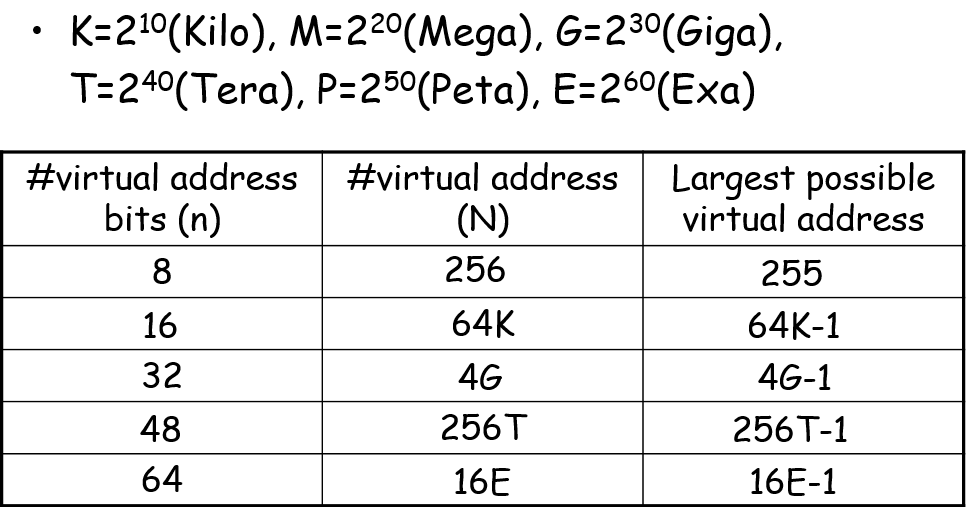
****

**答案：信号发不出去，只能保持一个，第1个发送成功，2、3、4被覆盖，第5个保存，最后会接收到**

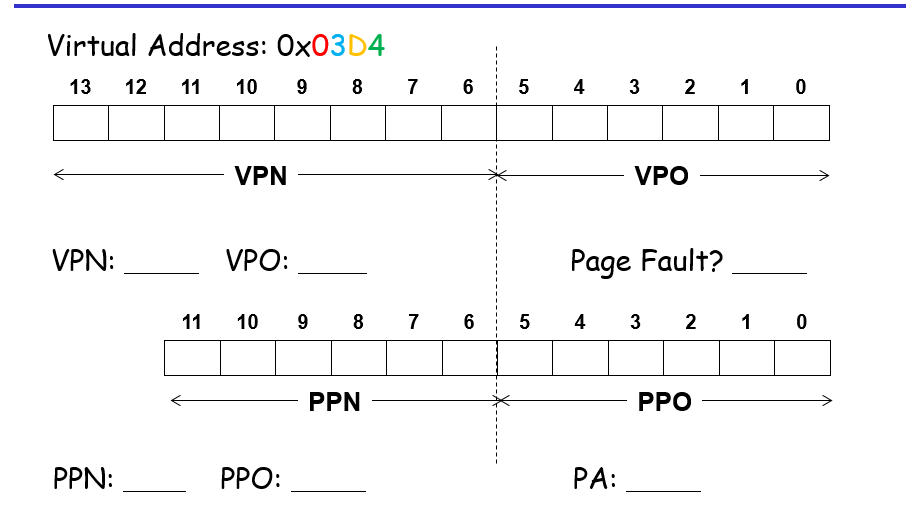
**第四章 虚拟化：虚拟内存**

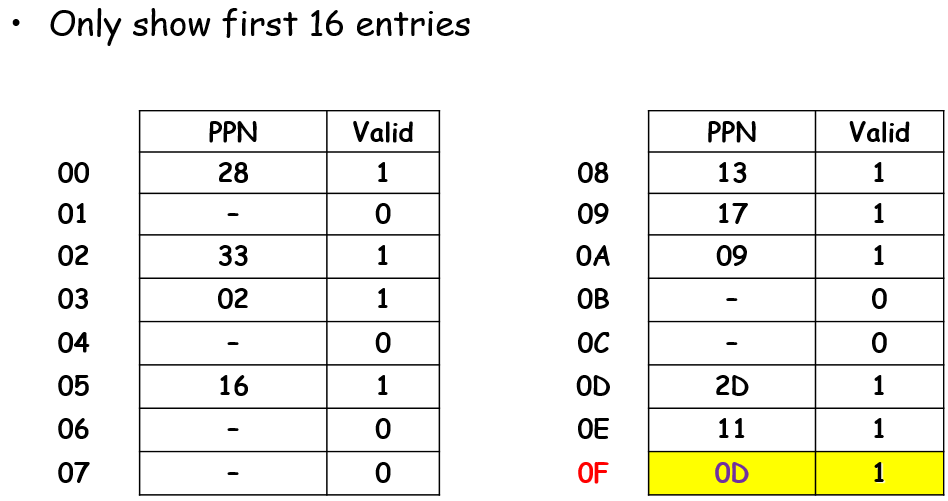


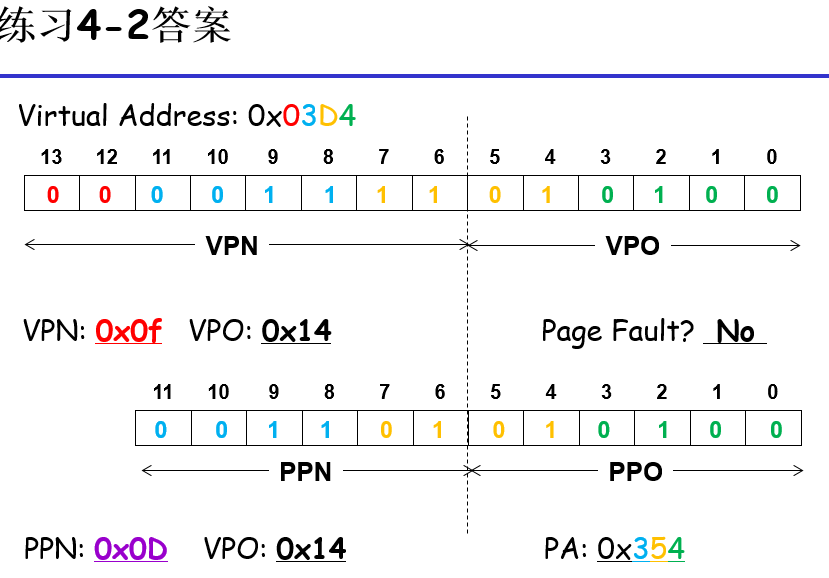
**答案：**



**二：**







**三：给定一个32位的虚拟地址空间和一个24位的物理地址，当页面大小P分为别1KB，2KB，4KB，8KB时，请确定VPN、VPO、PPN、PPO的位数。**

**答案：P=1KB**

**VPN=22; VPO=10; PPN=14; PPO=10**

**P=2KB**

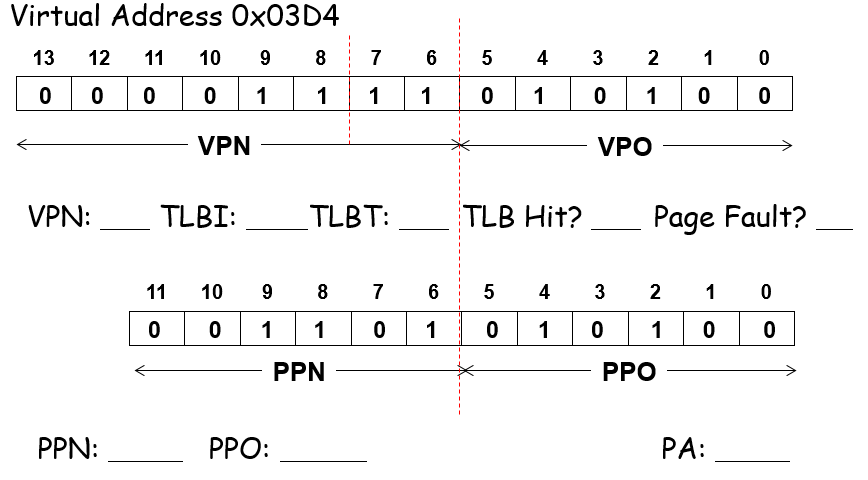
**VPN=21; VPO=11; PPN=13; PPO=11**

**P=4KB**

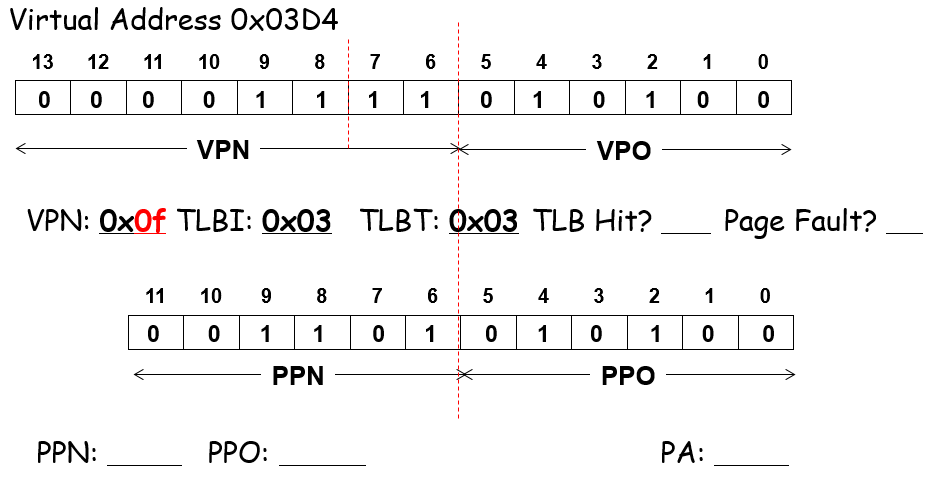
**VPN=20; VPO=12; PPN=12; PPO=12**

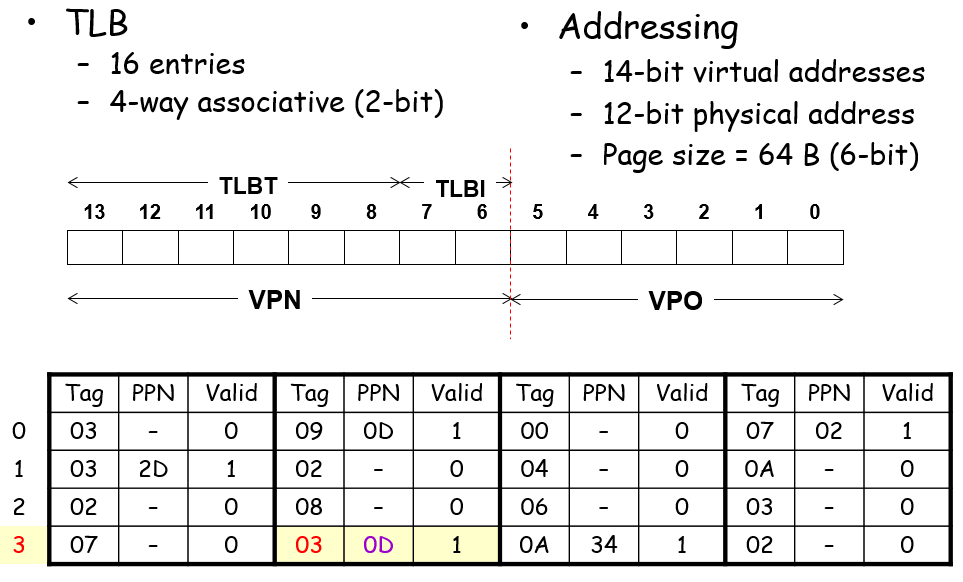
**P=8KB**

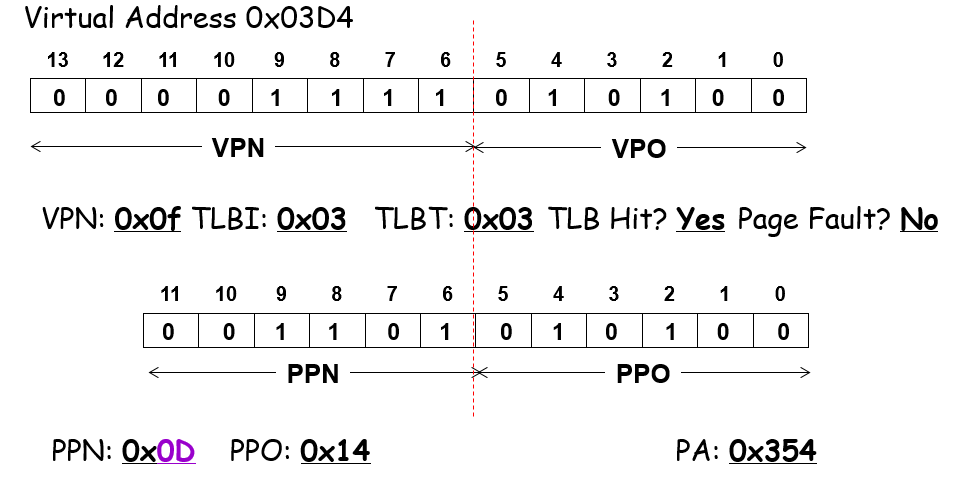
**VPN=19; VPO=13; PPN=11; PPO=13**

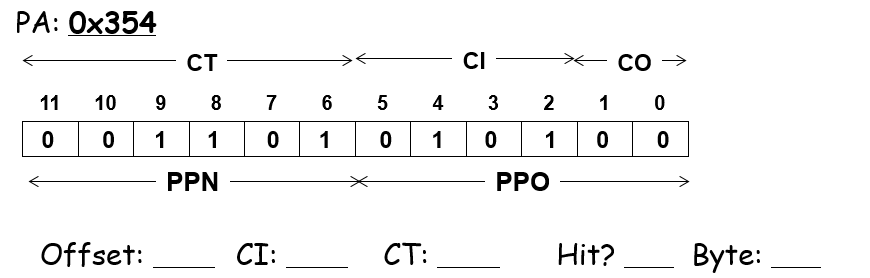
**四：**

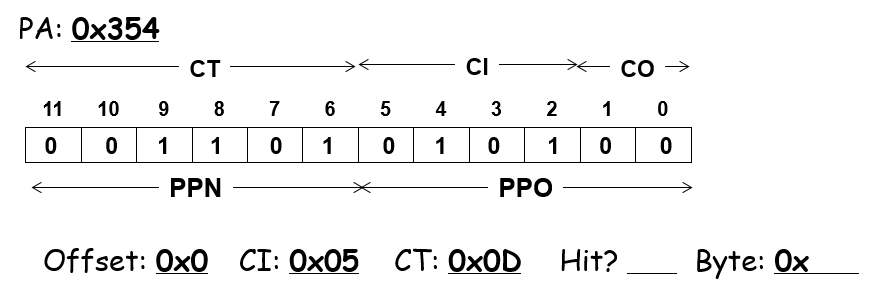
**答案：**



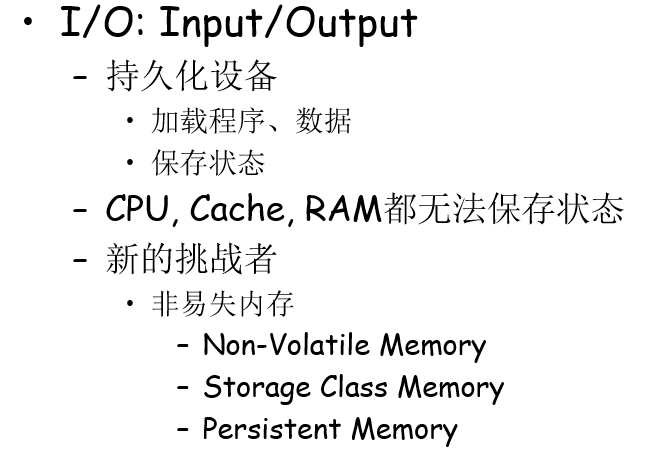


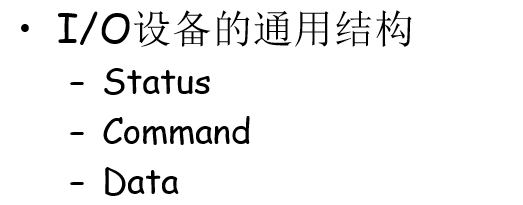


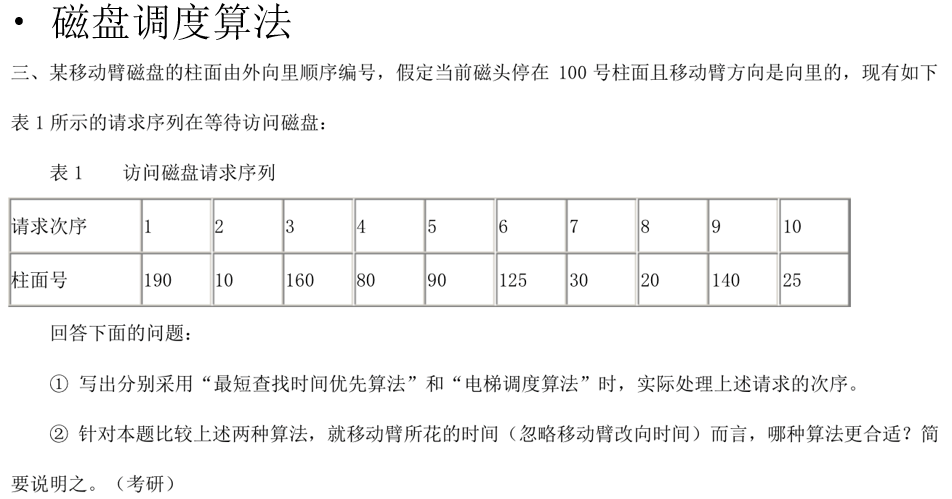
**五：**



**第五章 持久化：文件系统**







**答案：最短查找时间优先**

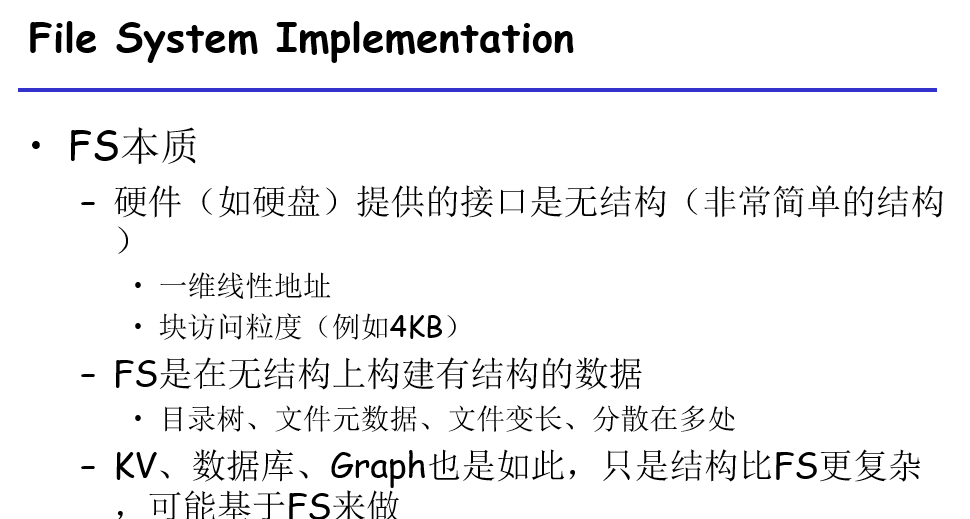
**100-90-80-125-140-160-190-30-20-10**

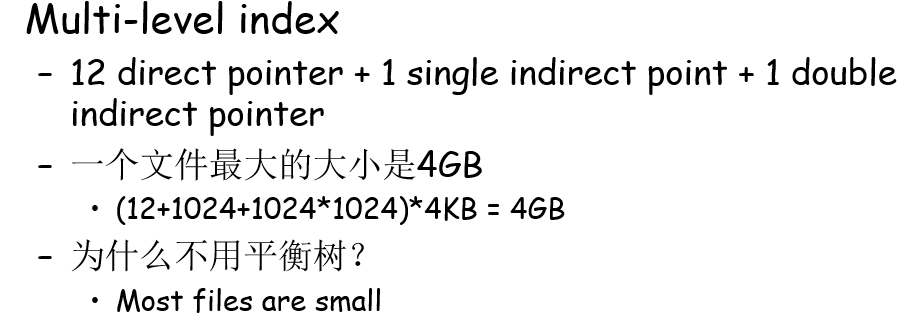
**10+10+45+15+20+30+160+10+10 = 310**

**电梯算法（F-SCAN)**

**100-90-80-30-20-10-125-140-160-190**

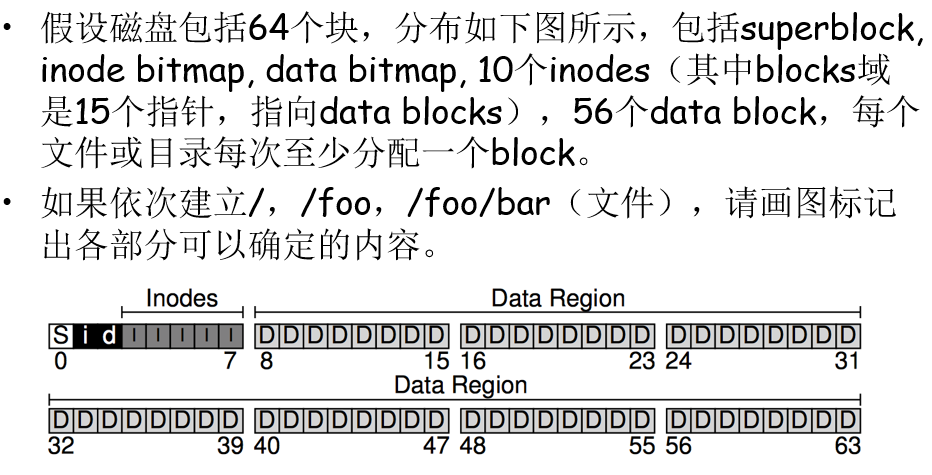
**10+10+50+10+10+115+15+20+30 = 270**







**二：**



**三：假设32位计算机中，每个data block的大小是4KB，inode结构中包括12个direct pointers，2个indirect point，1个double indirect pointer。**

**请问该文件系统支持的一个文件最大为多大？**

**假设一个文件是100KB，请画出这些指针的结构。**

**四：文件系统为保持数据一致性，有哪些主要方法，各自的主要原理和特点是什么？**

**（至少列出3个）**

**Solution #1: The File System Checker**

**目标：让元数据保持一致，在文件系统被mount之前检查**

**Fsck检查什么？**

**Superblock: 检查superblock看起来是不是正常**

**Free blocks: 检查inode, indirect blocks, double indirect blocks, 看数据在磁盘上如何分布。然后基于这些信息去纠正bitmap；同样去纠正inode bitmap**

**Inode state: 检查inode的每个域是否正常；如果错误纠正不了，就会删除inode，同时更新inode bitmap**

**Inode links: 不同的目录包括同一个文件(link)，扫描整个目录树；如果一个文件不被任何一个目录包括，则放到lost+found目录**

**Deduplicates: 检查deduplicate pointers，两个不同的Inode指向同一个block，检查Inode是不是bad，block是否重复**

**Bad Blocks: 检查block pointers是否有问题，指向内容是否出界？如果出界只是去掉bad pointers或indirect blocks**

**Directory Check: 确保.和..是每个目录最开始的两项，每一项都有对应的inode，不会有目录被链接两次**

**特点：FSCK最大的问题：太慢了！只能整盘扫描，例如只修改3个block，扫描整盘效率很低**

**Solution #2: Journaling (Write-Ahead Logging)**

**Basic Idea**

**写数据之前，先写到log里面，描述要做什么**

**如果写入数据过程中系统崩溃，在恢复时可以依据log的内容回滚，然后重新尝试**

**如果写日志过程中崩溃，那么就当没发生过，不会出现数据不一致或写一半的情况**

**避免扫描整个磁盘，只需查看日志区（完成的日志会删除，所以日志区不大）**

**总结：**

**1. Journal Write 2. Journal Commit 3. Checkpoint**

**进一步改进：**

**改进：增加checksum，这样可以一起写入，Linux ext4采用**

**Solution #3: Other Approaches**

**1）Copy-On-Write (COW)**

**Used in Sun’s ZFS**

**永远不原位更新，写到新的位置上**

**更新数量达到阈值，修改root指针，指向更新后的数据**

**FS一直保持一致性**

**LFS是COW的一个早期实现版本**

**2）Backpointer-Based Consistency (BBC)**

**数据块增加back pointer，回头看inode等元数据的forward pointer是否指向自己，从而判断一致性**

**第六章：并发**

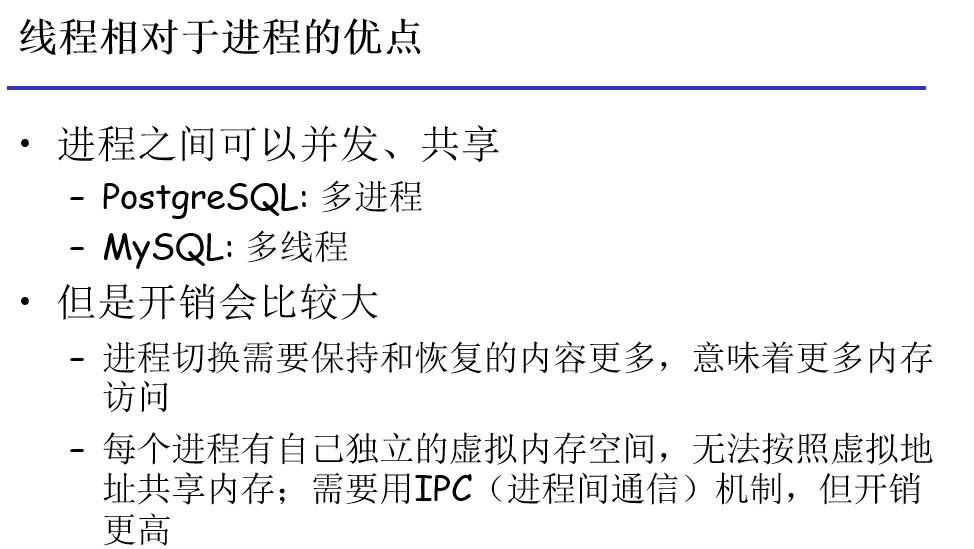
**A process with multiple threads：**

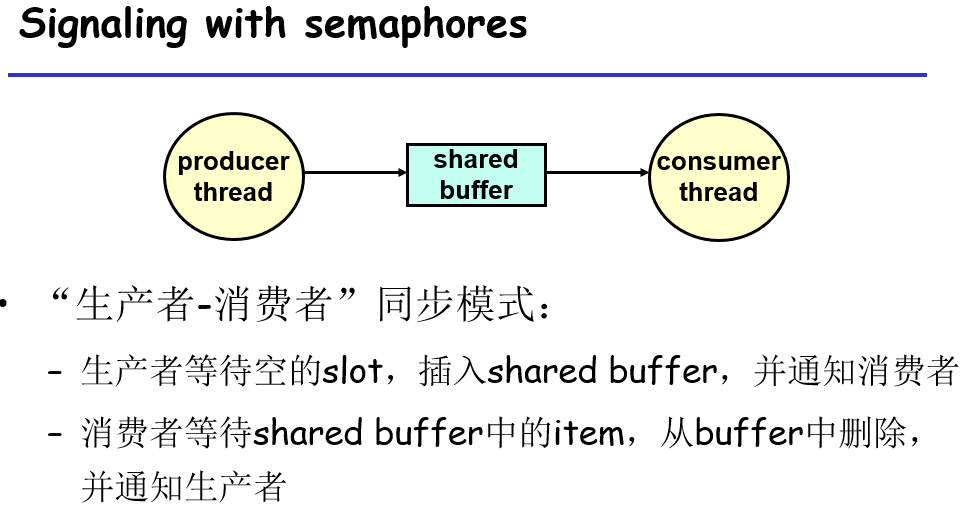
**多个线程可以共享同一个进程中的部分资源**

**每个线程有自己的流程控制（PC的值，寄存器的值，栈）**

**每个线程共享相同的code, data和kernel context**

**每个进程有自己的thread id (TID)**





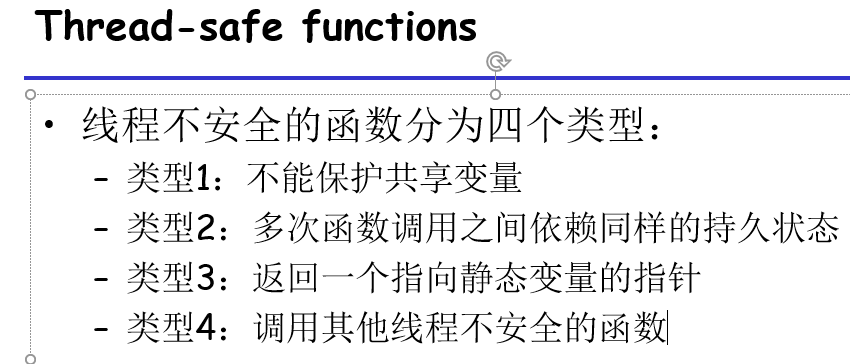
**6.1对于读写者问题，ppt中给出的是读优先，例如**

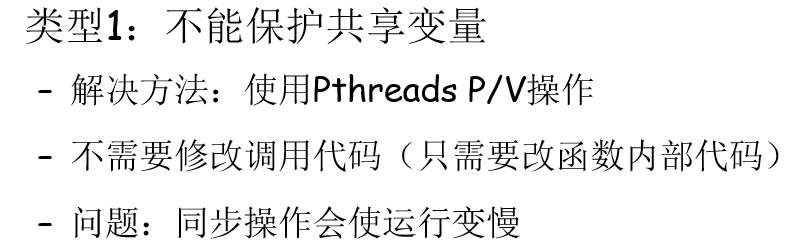
**请求序列：R1, W1, R2, R3, W2**

**如果R2在R1结束前到达，执行顺序是R1,R2,R3,W1,W2，即后来的R会先于W执行**

**A. 如果最多只有N个读者，能否修改程序让读写者尽可能公平？(正在W，到来的R,W竞争；正在R，到来的R,W竞争)**

**B. 如何实现一个写者优先级高于读者的程序？（正在W，到来的W优先；正在R，新来的按顺序调度）**





**二：**

