操作系统强化课考试

某天，王道考研自习室楼下新开了一家餐厅——楼楼手作寿司店。这家店非常上流，有 n 位寿司师傅为客人提供一对一服务，现场做料理。一位客人到店时，需要先取号，并等待叫号。没有客人的时候，寿司师傅可以睡觉休息。有客人的时候，只要有空闲的寿司师傅，就叫号，让下一位客人进店就餐，并由寿司师傅现场做料理。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

```
int waiting=0;           //等候服务的顾客数
semaphore mutex=1;       //互斥访问waiting
semaphore customer=0;    //顾客资源
semaphore =0;            //服务员资源

server_i(){              //n个服务员进程
  while(1){
    P(customers);        //开始干活前，要消耗一个“顾客资源”，若无顾客，则睡眠休息
    P(mutex);            //互斥访问 waiting 变量
    waiting--;           //等候顾客数少一个，相当于“叫号”，可根据题目场景写上中文描述
    V(server);           //释放一个“服务员资源”
    V(mutex);            //开放对 waiting 变量的锁
    提供服务;            //根据题目场景给出中文描述
  }
}
customer(){          //顾客进程
  P(mutex);          //进程互斥
  waiting++;         //等候顾客数加1,相当于“取号”
  V(customers);      //释放一个“顾客资源”
  V(mutex);          //开放临界区
  P(server);         //被服务前，要消耗一个“服务员资源”，若无空闲服务员，则阻塞等待
  被服务;            //根据题目场景给出中文描述
}
```

某天，王道考研自习室楼下新开了一家餐厅——楼楼手作寿司店。这家店非常上流，有 n 位寿司师傅为客人提供一对一服务，现场做料理。一位客人到店时，需要先取号，并等待叫号。这家店老板很黑心，没有客人的时候，寿司师傅也不可以睡觉休息，必须“忙等”。有客人的时候，只要有空闲的寿司师傅，就叫号，让下一位客人进店就餐，并由寿司师傅现场做料理。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

```
int waiting=0;           //等候服务的顾客数
semaphore mutex=1;       //互斥访问waiting
semaphore customer=0;    //顾客资源
semaphore =0;            //服务员资源

server_i(){              //n个服务员进程
  while(1){
    P(mutex);              //互斥访问 waiting 变量
    if(waiting > 0){       //有顾客
      waiting--;           //等候顾客数少一个，相当于“叫号”，可根据题目场景写上中文描述
      V(server);           //释放一个“服务员资源”
      V(mutex);            //开放对 waiting 变量的锁
```

```
    } else {                //没有顾客
      V(mutex);             //开放对 waiting 变量的锁
      continue;             //循环检查是否有顾客，“忙等”
    }
    提供服务;               //根据题目场景给出中文描述
  }
}
customer(){               //顾客进程
  P(mutex);               //进程互斥
  waiting++;              //等候顾客数加1,相当于“取号”
  V(customers);           //释放一个“顾客资源”
  V(mutex);               //开放临界区
  P(server);              //被服务前，要消耗一个“服务员资源”，若无空闲服务员，则阻塞等待
  被服务;                 //根据题目场景给出中文描述
}
```

某天，王道考研自习室楼下新开了一家餐厅——楼楼手作寿司店。这家店非常上流，有 n 位寿司师傅为客人提供一对一服务，现场做料理。为了营造“这家店很火”的感觉，老板在店门口摆了m个等位座椅供客人等位使用。一位客人到店时，会先观察还有没有等位座椅，如果没有座椅可用，就转身离开；如果有座椅可用，就会先取号，并坐下等待叫号。没有客人的时候，寿司师傅可以睡觉休息。有客人的时候，只要有空闲的寿司师傅，就叫号，让下一位客人进店就餐，并由寿司师傅现场做料理。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

```
int waiting=0;            //等候服务的顾客数
semaphore mutex=1;        //互斥访问waiting
semaphore customer=0;     //顾客资源
semaphore =0;             //服务员资源

server_i(){               //n个服务员进程
  while(1){
    P(customers);         //开始干活前，要消耗一个“顾客资源”，若无顾客，则睡眠休息
    P(mutex);             //互斥访问 waiting 变量
    waiting--;            //等候顾客数少一个，相当于“叫号”，可根据题目场景写上中文描述
    V(server);            //释放一个“服务员资源”
    V(mutex);             //开放对 waiting 变量的锁
    提供服务;             //根据题目场景给出中文描述
  }
}
customer(){               //顾客进程
  P(mutex);               //进程互斥
  if(waiting<m){          //若有空的椅子，就找到椅子坐下等待
    waiting++;            //等候顾客数加1,相当于“取号”
    V(customers);         //释放一个“顾客资源”
    V(mutex);             //开放临界区
    P(server);            //被服务前，要消耗一个“服务员资源”，若无空闲服务员，则阻塞等待
    被服务;               //根据题目场景给出中文描述
```

```
  } else {
    V(mutex);            //人满，离开
  }
}
```

某男子足球俱乐部，有教练、队员若干。每次足球训练开始之前，教练、球员都需要先进入更衣室换衣服，可惜俱乐部只有一个更衣室。教练们脸皮薄，无法接受和别人共用更衣室。队员们脸皮厚，可以和其他队员一起使用更衣室。如果队员和教练都要使用更衣室，则应该让教练优先。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

【参考答案】本题中，教练就是写者，队员就是读者，不过是读者写者问题换了个马甲而已。按照题目要求，要求实现“写优先”。注意和王道书上的两种“读者-写者”解法对比学习。“写优先”代码如下：

```
semophore read=1;                //互斥信号量，用于给读者“上锁”
semophore write=1;               //互斥信号量，用于给写者“上锁”
semophore rmutex=1;              //互斥信号量，实现对readCount的互斥访问
semophore wmutex=1;              //互斥信号量，实现对writeCount的互斥访问
int readCount=0, writeCount=0;   //读者、写者的数量

//读者进程（在这个题里就是可以多人一起共用更衣室的队员们）
Reader(){
  while(true){
   P(read);        //每个读者到达时先对 read 上锁
   P(rmutex);
     readCount++;
     if(readCount==1) P(write); //第一个开始读的读者对写者上锁
   V(rmutex);
   V(read);        //每个读者正式开始读之前对 read 解锁
   读者读...;
   P(rmutex);
     readCount--;
     if(readCount==0) V(write); //最后一个读完的读者对写者解锁
   V(rmutex);
  }
}

//写者进程（在这个题目里，对应必须独享更衣室的教练们）
Writer(){
  while(true){
   P(wmutex);
     writeCount++;
     if(writeCount==1) P(read);    //第一个到达的写者对读者上锁，这一步是实现“写优先”的关键
   V(wmutex);
   P(write);     //每个写者开始写之前都要 P(write)，保证写者之间互斥，同时也能保证若当前有读者正在读，那么写者等待
   写者写...;
   V(write);
   P(wmutex);
```

```
        writeCount--;
        if(writeCount==0) V(read); //最后一个写者写完之后，对读者解锁
    V(wmutex);
   }
}
```

为了方便大家对比学习，下面再附上“读者优先”的实现，这种方式可能导致写者饥饿：

```
semaphore rw=1; //用于实现对共享文件的互斥访问
int count = 0;  //记录当前有几个读进程在访问文件
semaphore mutex = 1;  //用于保证对count变量的互斥访问

writer (){
while(1){
   P(rw);   //写之前“加锁”
   写文件…
   V(rw);   //写完了“解锁”
}
}

reader (){
while(1){
   P(mutex);  //各读进程互斥访问count
   if(count==0) //由第一个读进程负责
       P(rw); //读之前“加锁”
   count++; //访问文件的读进程数+1
   V(mutex);
   读文件…
   P(mutex);  //各读进程互斥访问count
   count--; //访问文件的读进程数-1
   if(count==0) //由最后一个读进程负责
       V(rw); //读完了“解锁”
   V(mutex);
}
}
```

下面是“读写公平法”。也就是王道书里的第二种方法。

```
semaphore rw=1;         //用于实现对共享文件的互斥访问
int count = 0;          //记录当前有几个读进程在访问文件
semaphore mutex = 1;    //用于保证对count变量的互斥访问
semaphore queue = 1;    //用于实现“读写公平”。咸鱼注：可以将queue理解为一个“队列”，当资源暂不可访问时，无论读者、写者都需要公平排队

writer (){
  while(1){
     P(queue);
     P(rw);
```

```
        写文件…
        V(rw);
        V(queue);
    }
}

reader (){
    while(1){
        P(queue);
        P(mutex);
        if(count==0)
            P(rw);
        count++;
        V(mutex);
        V(queue);
        读文件…
        P(mutex);
        count--;
        if(count==0)
            V(rw);
        V(mutex);
    }
}
```

对于上面三种解法，如果弄不清楚它们之间的区别，不妨带入一个例子看看。假设每个读者的读操作都耗时较长，读者写者到达的顺序是：

读者1——读者2——读者3——写者A——读者4——写者B——读者5

如果采用“读者优先”的实现方法，那情况是这样的：读者1到达并开始读，紧接着读者2、读者3到达，都可以开始读；写者A到达，暂时不能写；读者4到达，可以开始读；写者B到达，暂时不能写；读者5到达，可以直接开始读；等读者1、2、3、4、5都读完之后，写者A、写者B才可以依次进行写。

如果采用“读写公平法”的实现方法，那情况是这样的：读者1到达并开始读，紧接着读者2、读者3到达，都可以开始读；写者A到达，暂时不能写；读者4到达，暂时不能读；写者B到达，暂时不能写；读者5到达，暂时不能读；等读者1、2、3都读完之后，写者A开始写；写者A写完之后读者4开始读；读者4读完后写者B开始写；写者B写完后读者5开始读。

如果采用“写优先”的实现方法，那情况是这样的：读者1到达并开始读，紧接着读者2、读者3到达，都可以开始读；写者A到达，暂时不能写；读者4到达，暂时不能读；写者B到达，暂时不能写；读者5到达，暂时不能读；等读者1、2、3都读完之后，写者A开始写；写者A写完之后写者B开始写；写者B写完后读者4开始读，同时读者5也可以开始读。

俗话说，“干饭人，干饭魂，干饭人吃饭得用盆”。一荤、一素、一汤、一米饭，是每个干饭人的标配。饭点到了，很多干饭人奔向食堂。每个干饭人进入食堂后，需要做这些事：拿一个盆打荤菜，再拿一个盆打素菜，再拿一个盆打汤，再拿一个盆打饭，然后找一个座位坐下干饭，干完饭把盆还给食堂，然后跑路。现在，食堂里共有N个盆，M个座位。请使用P、V操作描述上述过程的互斥与同步，并说明所用信号量及初值的含义。

参考答案：显然，这个题目的关键是不能发生死锁。死锁问题应该参考哲学家问题的解决思路。在哲学家进餐问题中，我们解决死锁的方法有三种：

1. 奇数号哲学家必须先拿左手的筷子，偶数号哲学家必须先拿右手的筷子
2. 限制“最多允许4个哲学家同时进餐”
3. 仅当一个哲学家左右两边的筷子都可用时才允许哲学家拿筷子

其中，第一种方法的思想很难迁移到其他题目中。但是第二、第三种思想可以迁移到大多数死锁题目。

我们先来看一个标准的错误解法：

```
semaphore pot=N;  //同步信号量，用于表示“盆”资源，食堂里总共有N个盆
semaphore seat=M; //同步信号量，用于表示“作为”资源，食堂里总共有M个座位

//干饭人进程
EatMan(){
  进食堂;
  P(pot);   //拿一个盆
  打荤菜;
  P(pot);   //拿一个盆
  打素材;
  P(pot);   //拿一个盆
  打汤;
  P(pot);   //拿一个盆
  打饭;
  P(seat);  //占个座
  干饭;
  V(seat);  //让出座位
  V(pot);   //归还干饭盆
  V(pot);
  V(pot);
  V(pot);
  离开食堂;
}
```

显然，上面这种解法会导致死锁。假设同时来了好多个干饭人，每个人都拿三个盆，盆很快就会被拿光。那所有人都无法得到第四个盆，就会发生死锁。

下面我们模仿哲学家进餐问题的第二种解决思路。在哲学家问题中，共有5个哲学家，如果我们限制“最多允许4个哲学家同时进餐”，那么至少会有一个哲学家可以同时获得左右两只筷子，并顺利进餐，从而预防了死锁。

同样的思路可以迁移到干饭人问题中。每个干饭人需要同时持有4个盆才能干饭，那么最糟糕的情况是每个干饭人都持有3个盆，同时在等待第四个盆。此时，但凡再多一个盆，就至少能有一个干饭人可以顺利干饭，就不会死锁。因此我们可以限制同时抢盆的人数为 x，那么只要满足 $3x + 1 \le N$，则一定不会发生死锁，可得 $x \le (N-1)/3$。参考代码如下：

```
semaphore pot=N;      //同步信号量，用于表示“盆”资源，食堂里总共有N个盆
```

```
semaphore seat=M;      //同步信号量，用于表示“作为”资源，食堂里总共有M个座位
semaphore x=(N-1)/3;   //同步信号量x，用于表示最多允许多少个人同时干饭。(N-1)/3 向下取整
//干饭人进程
EatMan(){
  P(x);      //进食堂拿盆之前，先看看是否已到达人数上限
  进食堂;
  P(pot);    //拿一个盆
  打荤菜;
  P(pot);    //拿一个盆
  打素材;
  P(pot);    //拿一个盆
  打汤;
  P(pot);    //拿一个盆
  打饭;
  P(seat);   //占个座
  干饭;
  V(seat);   //让出座位
  V(pot);    //归还干饭盆
  V(pot);
  V(pot);
  V(pot);
  离开食堂;
}
```

上面这种做法，限制了人数上限，且先拿盆，再占座，一定不会发生死锁。当然，如果先占座、后拿盆，也不会死锁。事实上，如果座位的数量满足 seat ≤ (N-1)/3，那么甚至可以不设置专门的信号量x，完全可以先占座，后拿盆，也一定不会死锁。因为座位的数量就可以限制同时抢盆的人数。

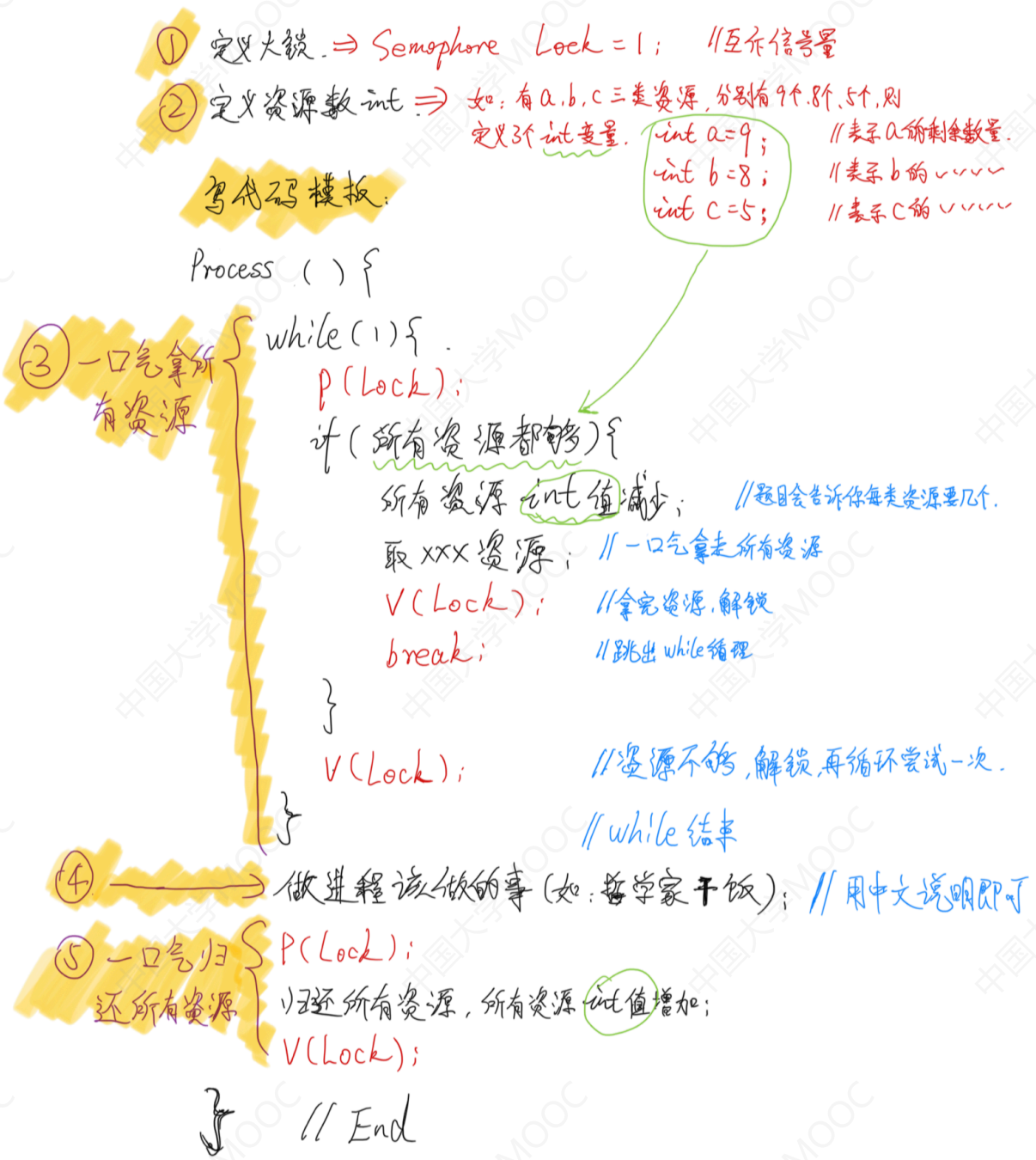
下面我们再模仿哲学家问题的第三种解决思路——仅当一个哲学家左右两边的筷子都可用时才允许哲学家拿筷子。破坏了“请求和保持”条件，采用“静态分配”的思想，让进程一口气获得所有资源，再开始运行。代码如下：

```
semaphore mutex=1;       //互斥信号量，保证所有进程对 pot 变量、seat 变量的访问是互斥的。
semaphore pot=N;         //用于表示“盆”资源，食堂里总共有N个盆
semaphore seat=M;        //用于表示“作为”资源，食堂里总共有M个座位

//干饭人进程
EatMan(){
  进食堂;
  P(mutex);
  P(pot);    //一口气拿四个盆，并占座
  P(pot);
  P(pot);
  P(pot);
  P(seat);
  V(mutex);

  打荤菜;
  打素材;
```

```
    打汤;
    打饭;
    干饭;

    V(seat);  //让出座位
    V(pot);   //归还干饭盆
    V(pot);
    V(pot);
    V(pot);
    离开食堂;
}
```

这个题目想告诉大家的是，哲学家进餐问题的解决思路中，后两种方法更为通用，可以作为考试时主要的策略。大家再思考一下，限制人数上限、一口气拿所有资源，哪种方案的并发度更高一些呢？显然是后者对吧。“限制人数上限”的方案中，最糟糕的情况是，只有一个人获得了4个盆，其余进程都只有3个盆，也就是说只有1个进程可以顺利运行下去，因此并发度低。

最后，再使用强化课P2给大家的模板，也就是用int变量表示资源




```
semaphore mutex=1;       //互斥信号量，保证所有进程对 pot 变量、seat 变量的访问是互斥的。
int pot=N;               //用于表示“盆”资源，食堂里总共有N个盆
int seat=M;              //用于表示“作为”资源，食堂里总共有M个座位

//干饭人进程
EatMan(){
  进食堂;

  while(1){
    P(mutex);    //资源上锁
    if ( pot>=4 && seat>=1 ) {   //所有资源都够
      pot -=4;   //一口气拿走所有资源
      seat -=1;
      V(mutex); //拿完资源，解锁
```

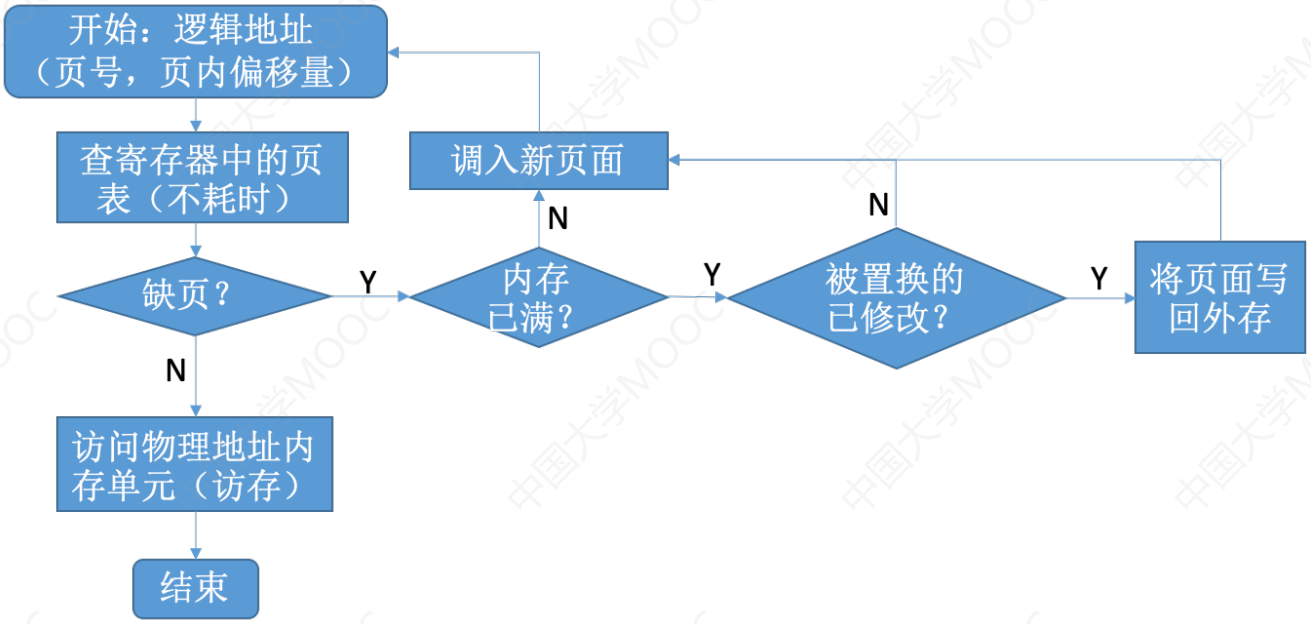
```
        break;     //拿完资源，跳出循环
      }
      V(mutex);    //资源不够，解锁
    } //while

    打荤菜;
    打素材;
    打汤;
    打饭;
    干饭;

    P(mutex);    //资源上锁
    pot +=4;     //一口气归还所有资源
    seat +=1;
    V(mutex);    //资源解锁

    离开食堂;
}
```

现有一请求页式系统，页表保存在寄存器中，查页表几乎不耗时。若有一个可用的空页或被置换的页未被修改，则它处理一个缺页中断需要8ms；若被置换的页已被修改，则处理一缺页中断因增加写回外存时间而需要20ms，内存的存取时间为1ms。

1）该系统的页表项的中，需要包含哪些信息?

2）发生缺页时，70%的概率需要置换一个被修改的页面，为保证有效存取时间不超过2ms，可接受的最大缺页中断率是多少?

> 参考答案：
>
> 1）根据题目条件，该请求分页系统的页表项中，至少应该包含以下信息：
>
> - 页号（隐含，实际不会占用存储空间）
> - 页框号（用于描述逻辑页面在物理内存中的位置）
> - 脏位（用于描述该页面的数据是否被修改过）
> - 外存地址（用于描述该页面在外存中的存放地址）
> - 置换算法相关的信息（不同的置换算法，需要记录的信息不同。如FIFO算法需要记录页面调入内存的时间，LRU算法需要记录页面最近一次被访问的时间）
>
> 2）该系统中，访问一个逻辑地址的过程如下图所示：



假设缺页率为 p，则：

- 若未缺页，只需查寄存器中的页表（不耗时），然后访问目标内存单元（耗时1ms），总耗时 1 ms
- 若缺页，且内存未满，或内存已满但被置换的页面未修改，则总耗时 8ms + 1 ms
- 若缺页，且内存已满，被置换的页面已修改，则总耗时 20ms + 1 ms

要保证有效存取时间不超过2ms，缺页率 p 需要满足下述条件：

$1*(1-p) + (8+1)*(30\% * p) + (20+1)*(70\%*p) \le 2$

即 $p \le 0.06$

最大缺页中断率为 6%

已知系统为32位实地址，采用36位虚拟地址，页面大小4KB，页表项大小为8B。

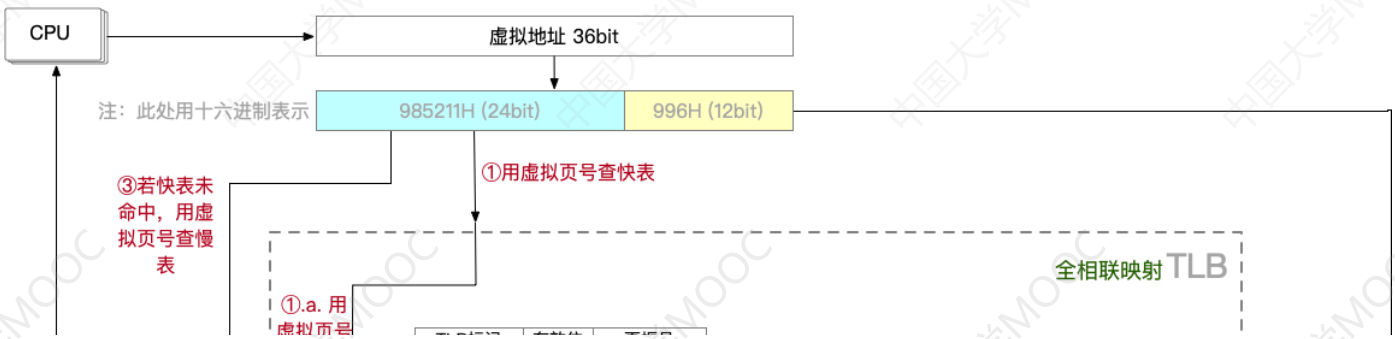
1）假设系统采用一级页表，TLB命中率为98%，TLB访问时间10ns，内存访问时间100ns，并假设当TLB访问失败时才开始访问内存，则平均的地址转换时间是多少?

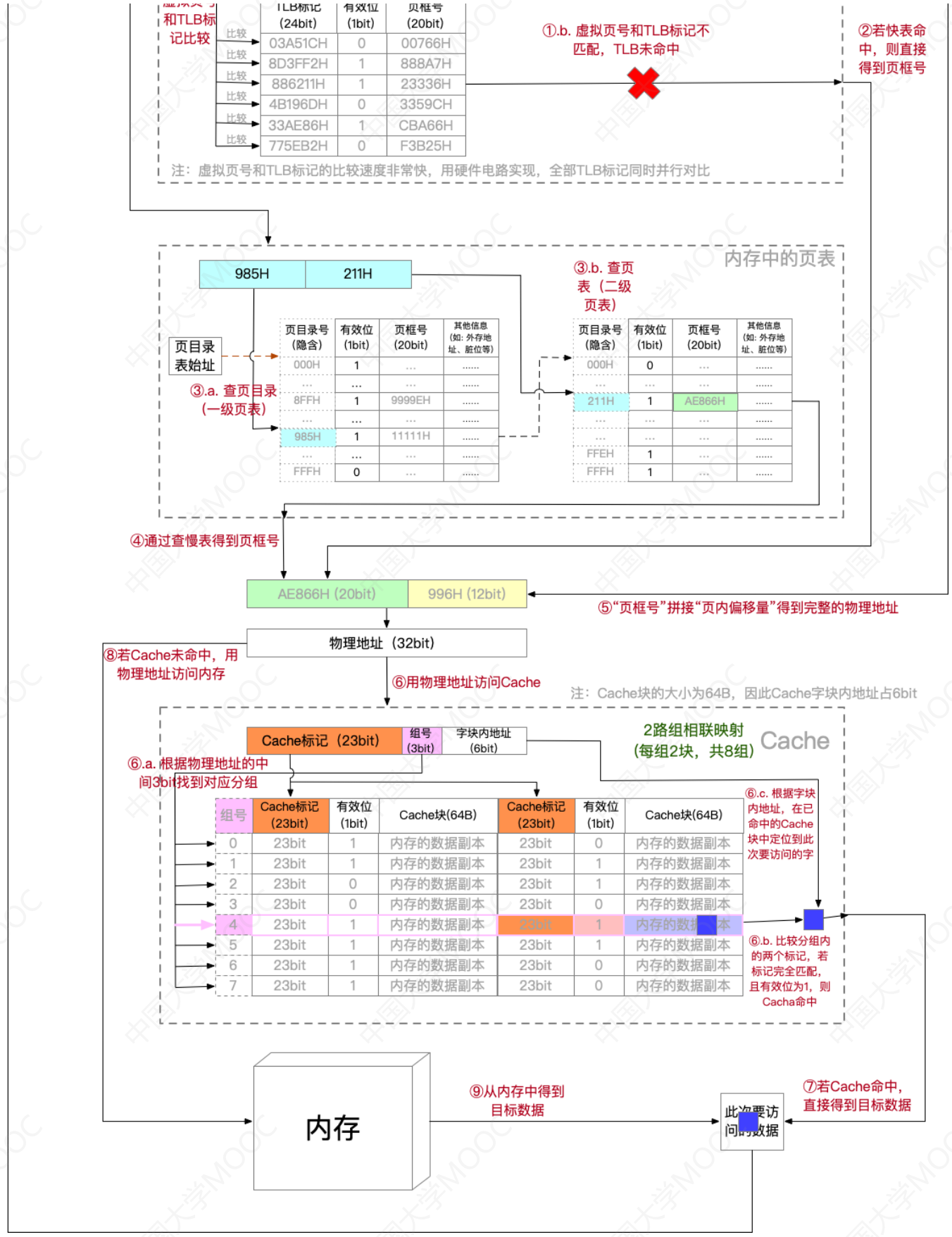
2）如果是二级页表，则平均的地址转换时间是多少?

3）上题中，如果要满足平均地址转换时间小于120ns，那么命中率需要至少多少?

4）指出下面这个图中，画的不合理的地方

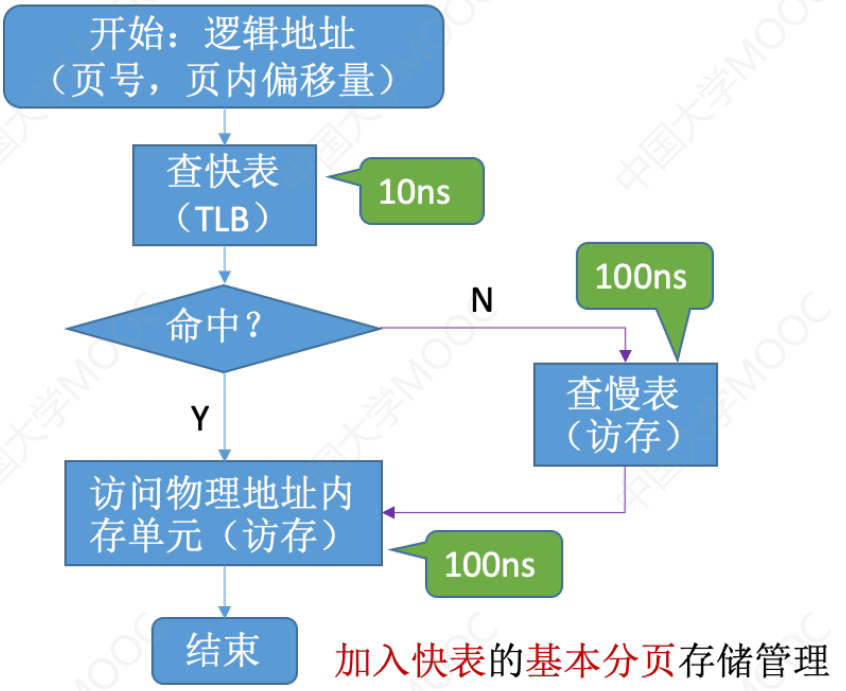
假设：某36位系统，按字节编制，每个页面大小为 4KB，则页内偏移量占 12 bit，虚拟页号24bit。物理地址空间大小为 4GB，因此物理地址共32bit，前 20bit 表示物理页框号





参考答案：

1）采用一级页表，访问一个虚拟地址的流程如下：

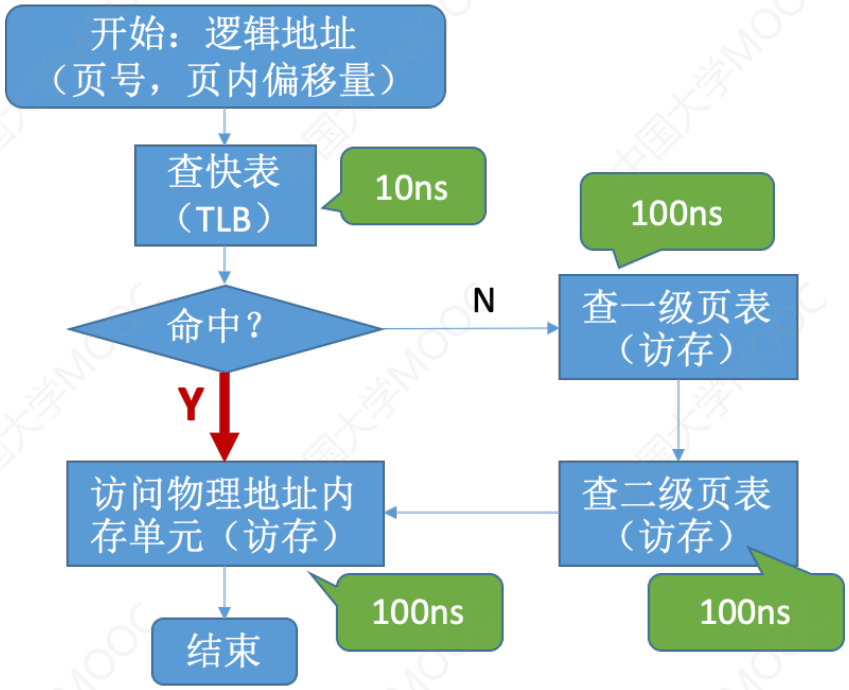


加入快表的基本分页存储管理

注意，题目问的是“地址转换时间”，而不是访问一个虚拟地址花费的全部时间。因此，本题的计算中，不考虑最后一次访存。

平均地址转换时间= 0.98 * 10 + 0.02 * (10+100) = 12ns

2）采用二级页表，访问一个虚拟地址的流程如下：



平均地址转换时间= 0.98 * 10 + 0.02 * (10+100+100) = 14ns

3）假设TLB命中率为p，则平均地址转换时间 = p * 10 + (1-p) * (10+100+100) ≤ 120ns

p ≥ 0.45，即TLB命中率不能低于45%

> 4）第四小问的图是存储系统骚图（可在操作系统强化课P3下载），该图的虚拟地址位数、物理地址位数、页面大小都是和本题一致的。但是在图示中，我们画的两级页表，每一级的页号有12bit，也就是说，每一级的页表包含 4K个页表项。然鹅，每个页面的大小为4KB，在多级页表中，一般每一级页表的大小不能超过一个页框的容量，因此这个图画的不太严谨。

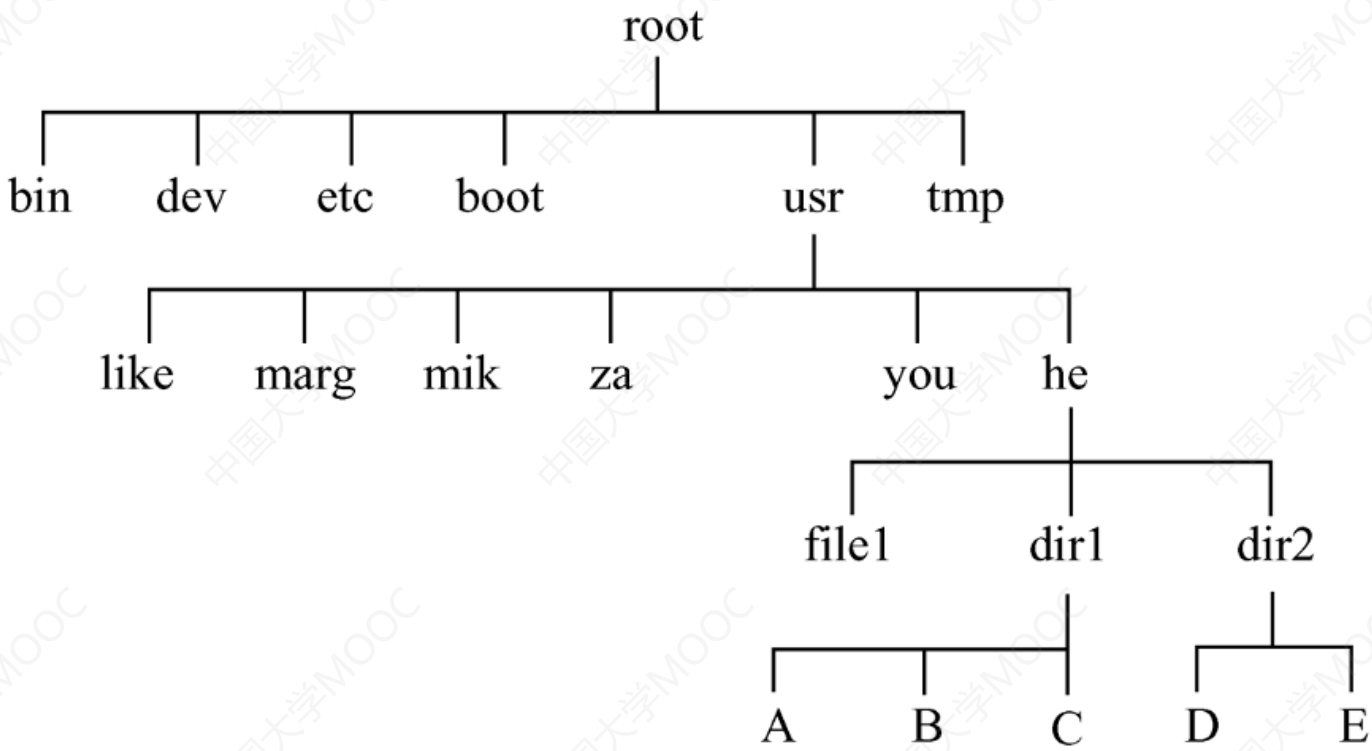
某个文件系统中，外存为硬盘。物理块大小为512B，有文件A包含598个记录，每个记录占255B，每个物理块放2个记录。文件A所在的目录如下图所示。

文件目录采用多级树形目录结构，由根目录结点、作为目录文件的中间结点和作为信息文件的树叶组成，每个目录项占127B，每个物理块放4个目录项，根目录的第一块常驻内存。试问：

1）若文件的物理结构采用链式存储方式，链指针地址占2B，那么要将文件A读入内存，至少需要存取几次硬盘?

2）若文件为连续文件，那么要读文件A的第487个记录至少要存取几次硬盘?

3）一般为减少读盘次数，可采取什么措施，此时可减少几次存取操作?



> 参考答案：见王道书 4.2_大题6。建议大家看看习题讲解的视频。
>
> 文件管理部分，题目条件多变、灵活。而王道书 4.1、4.2 的课后大题覆盖较为全面，建议大家尽量全做。

有一个文件系统如下图1所示。图中的方框表示目录，圆圈表示普通文件。根目录常驻内存，目录文件组织成链接文件，不设FCB，普通文件组织成索引文件。目录表指示下一级文件名及其磁盘地址（各占2B，共4B）。若下级文件是目录文件，指示其第一个磁盘块地址。若下级文件是普通文件，指示其FCB的磁盘地址。每个目录的文件磁盘块的最后4B供拉链使用。下级文件在上级目录文件中的次序在图中为从左至右。每个磁盘块有512B，与普通文件的一页等长。

普通文件的FCB组织如下图2所示。其中，每个磁盘地址占2B，前10个地址直接指示该文件前10页的地址。第11个地址指示一级索引表地址，一级索引表中每个磁盘地址指示一个文件页地址；第12个地址指示二级索引表地址，二级索引表中每个地址指示一个一级索引表地址；第13个地址指示三级索引表地址，三级索引表中每个地址指示一个二级索引表地址。请问：

1）一个普通文件最多可有多少个文件页?

2）若要读文件J中的某一页，最多启动磁盘多少次?

3）若要读文件W中的某一页，最少启动磁盘多少次?

4）根据3），为最大限度减少启动磁盘的次数，可采用什么方法？此时，磁盘最多启动多少次?

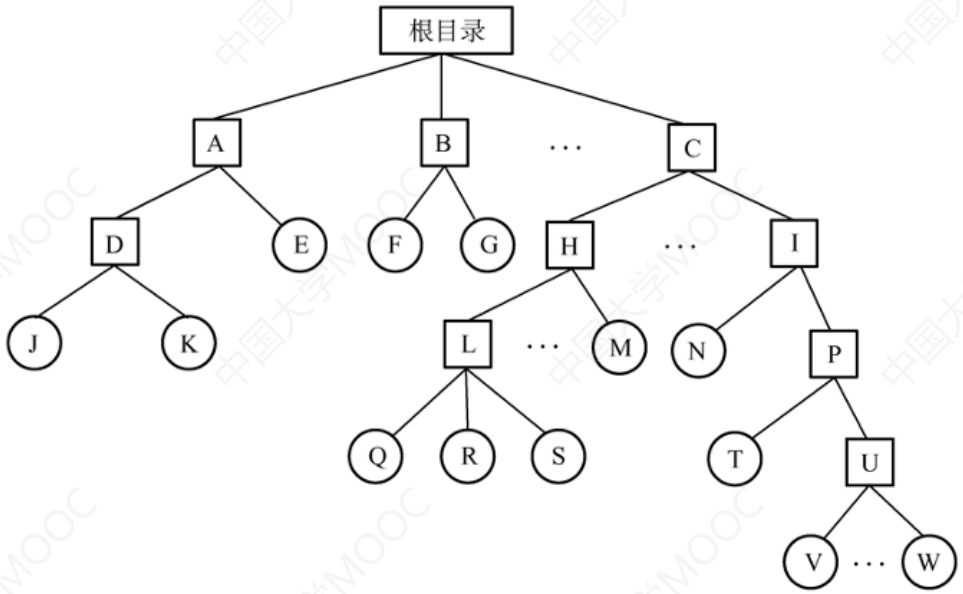


图 A　某树形结构文件系统框图

图 B　FCB 组织

> 参考答案：见王道书 4.2_大题5。建议大家看看习题讲解的视频。
>
> 文件管理部分，题目条件多变、灵活。而王道书 4.1、4.2 的课后大题覆盖较为全面，建议大家尽量全做。