一、乱七八糟题：

1.同步机制应遵循的规则：

1. 空闲让进。当无进程进入[临界区](https://www.baidu.com/s?wd=%E4%B8%B4%E7%95%8C%E5%8C%BA&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)时，相应的[临界资源](https://www.baidu.com/s?wd=%E4%B8%B4%E7%95%8C%E8%B5%84%E6%BA%90&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)处于空闲状态，因而允许一个请求进入[临界区](https://www.baidu.com/s?wd=%E4%B8%B4%E7%95%8C%E5%8C%BA&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的进程立即进入自己的[临界区](https://www.baidu.com/s?wd=%E4%B8%B4%E7%95%8C%E5%8C%BA&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)。
2. 忙则等待。当已有进程进入自己的临界区时，即相应的[临界资源](https://www.baidu.com/s?wd=%E4%B8%B4%E7%95%8C%E8%B5%84%E6%BA%90&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)正被访问，因而其它试图进入临界区的进程必须等待，以保证[进程互斥](https://www.baidu.com/s?wd=%E8%BF%9B%E7%A8%8B%E4%BA%92%E6%96%A5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)地访问临界资源。
3. 有限等待。对要求访问临界资源的进程，应保证进程能在有限时间进入临界区，以免陷入“死等”状态。
4. 让权等待。当进程不能进入自己的临界区时，应立即释放处理机，以免进程陷入忙等。
5. 消息传递通信的实现方式
6. 直接消息传递系统
7. 信箱通信

3.处理机调度层次

1. 高级调度：又称作业调度。调度对象是作业。主要任务是按一定的原则从外存上处于后备状态的作业中选择一个或多个，给它们创建进程、分配必要资源，并将它们放入就绪队列中
2. 低级调度：又称进程调度。调度对象是进程。主要任务是决定就绪队列中哪一个进程应获得处理机，并将处理机分配给它。
3. 中级调度：那些不能运行的进程，调至外存等待。它们已经具备运行条件且内存又稍有空闲时，把外存上那些已具备运行条件的就绪进程调入内存。

4.多级反馈队列调度算法

调度机制：

1. 设置多个就绪队列。并为每个队列赋予不同的优先级。第一个队列优先级最高，第二个次之。在优先级越高的队列中，其时间片就越小。
2. 每个队列都采用FCFS算法。
3. 按队列优先级调度。

调度算法的性能：

1. 终端型用户。
2. 短批处理作业用户。
3. 长批处理作业用户。

二、综合题：

1. 读者写着问题
2. FIFO、LRU

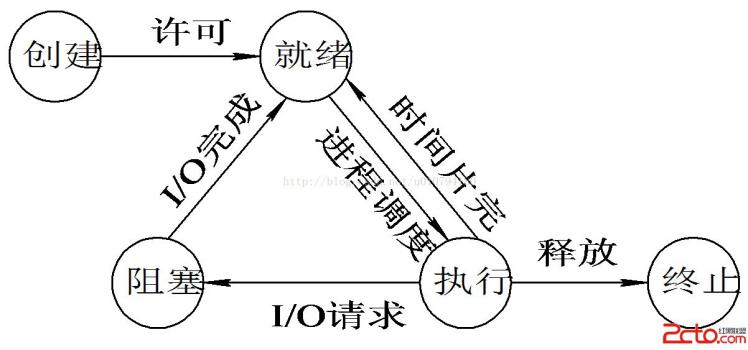
三、简述题：

1.进程的基本状态有哪些？这些状态之间是如何转换的？

进程的基本状态有：就绪，阻塞，执行三种。

执行到阻塞：I/O请求或等待事件发生

阻塞到就绪：I/O完成或事件已发生



2.简述银行家算法的基本原理

避免死锁。为实现银行家算法，每一个新进程进入系统之前必须声明运行过程中可能需要每种资源类型的最大单元数目。

1. 银行家算法中的数据结构

1）可利用资源向量Available

如果Available[j]=K，则表示系统中现有Rj类资源K个。

2）最大需求[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%93%B6%E8%A1%8C%E5%AE%B6%E7%AE%97%E6%B3%95/_blank)Max

如果Max[i,j]=K，则表示进程i需要Rj类资源的最大数目为K。

3）分配[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%93%B6%E8%A1%8C%E5%AE%B6%E7%AE%97%E6%B3%95/_blank)Allocation

如果Allocation[i,j]=K，则表示进程i当前已分得Rj类资源的 数目为K。

4）需求[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%93%B6%E8%A1%8C%E5%AE%B6%E7%AE%97%E6%B3%95/_blank)Need。

如果Need[i,j]=K，则表示进程i还需要Rj类资源K个，方能完成其任务。

Need[i,j]=Max[i,j]-Allocation[i,j]

1. 设Request是进程Pi的请求向量，如果Requesti[j]=K,表示进程Pi需要K个Rj类型的资源。当Pi发出资源请求后，系统按下述步骤进行检查：
2. 如果Requesti[j]<=Need[i,j],便转向步骤2）；否则认为出错，因为它所需要的资源数已超过它所宣布的最大值。
3. 如果Requesti[j]<=Available[j],便转向步骤3）；否则，表示尚无足够资源，Pi需等待。
4. 系统试探把资源分配给进程Pi，并修改下面数据结构中的数值：

Available[j]=Available[j]-Requesti[j]；

Allocation[i,j]= Allocation[i,j]+Requesti[j];

Need[i,j]=Need[i,j]-Requesti[j];

4)系统执行安全性算法，检查此次资源分配后系统是否处于安全状态。若安全，才正式将资源分配给进程Pi，以完成本次分配；否则，将本次的试探分配作废，恢复原来的资源分配状态，让进程Pi等待。

3.分页地址转换计算

逻辑地址转换为物理地址。给定一个逻辑地址空间中的地址为A，页面大小为L。

页号P=INT[A/L] //（INT是除整函数）

页内地址d=[A]MODL //（MOD是取余函数）

1. 计算页号和页内地址。
2. 找出对应页号块号，物理地址=对应页号块号\*L+d

4.产生死锁的必要条件和处理死锁的方法

必要条件（只有其中任何一个条件不成立，死锁就不会发生）：

1. 互斥条件：一个资源每次只能被一个进程使用。  
   （2） 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。  
   （3） 不可抢占条件:进程已获得的资源，在末使用完之前，不能被抢占。  
   （4） 循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系

处理死锁的方法：

（1）预防死锁。该方法是通过设置某些限制条件，去破坏产生死锁的四个必要条件中的一个或几个来预防产生死锁。

（2）避免死锁。同样是属于事先预防策略，在资源的动态分配过程中，用某种方法防止系统进入不安全状态，从而可以避免发生死锁。

（3）检测死锁。通过检测机构及时地检测出死锁的发生，然后采取适当的措施，把进程从死锁中解脱出来。

（4）解除死锁。当检测到系统中已发生死锁时，就采取相应的措施，把进程从死锁中解脱出来。常用的方法是撤消一些进程，回收它们的资源，将资源分配给已处于阻塞状态的进程，使其能继续运行。