1、考 处理器的三级调度 (多字数任务)

一个作业从提交开始直到完成，往往要经历三级调度。

高级调度（作业调度）High Scheduling

→运行频率较低，通常为几分钟一次

高级调度又称作业调度、宏观调度或长程调度。

其主要任务是按照一定原则从外存上处于后备状态的作业选择一个或者多个，给它们分配资源，建立相应进程。

其需要解决一下两个基本问题：

1. 接纳多少个作业
2. 接纳哪些作业

中级调度 Intermediate-level Scheduling

中级调度又称为中程调度或交换调度。

引入中级调度是为了提高内存利用率和系统吞吐量。

其主要任务是按照一定原则将处于外存对换区中的具备运行条件的进程调入内存，并将其修改为就绪状态，加入就绪队列；将处于内存中的暂时不能运行的进程交换到外存对换区；进程在外存交换区时的状态称挂起状态（分为挂起阻塞和挂起就绪）。

挂起

低级调度（进程调度）low level Scheduling

→运行频率很高，一般隔几毫秒运行一次。

低级调度又称微观调度、进程调度或短程调度。

其主要任务是按照一定原则从就绪队列中选取一个进程，将处理器分配给它。

2、考 调度算法的评价标准

周转时间 = 完成时间 - 到达时间

带权周转时间 = 周转时间 / 用时时间 (最小为1，越小越好)

3个时间点，4个时段

等待时间 运行时间

到达时间 被调时间 完成时间

周转时间

进程调度

进程调度的功能

进程调度的原因

不能进行进程调度的情况

抢占式

非抢占式

进程调度的方式

抢占方式、非抢占方式

高优先权优先调度算法
(High Priority scheduling)

基于时间片的轮转调度算法
(Round-Robin Scheduling)

2、常见调度算法

先来先服务调度算法
(First-Come, First-Served Scheduling)

短作业优先调度算法
(Shortest-Job-First Scheduling)

死锁

3、死锁的概念

当多个进程因竞争系统资源和相互通信而永久处于永久阻塞状态时，若无外力作用，这些进程都将无法向前推进。这些进程中的每一个进程，均无限期地等待其他进程占有的、自己无法获得的资源，这种现象称为死锁。

死锁有如下特征：

- 参与死锁的进程至少有两个。
- 每个参与死锁的进程均等待资源。
- 参与死锁的进程中至少有两个进程占有资源。
- 死锁进程是系统中当前进程集合的一个子集。

3、死锁产生的原因

OS中资源可分为可剥夺资源和不可剥夺资源。一个资源是否属于可剥夺资源，完全取决于资源本身的性质。

① 系统资源不足导致资源竞争

② 进程间推进顺序不当

进程在运行过程中，请求和释放资源的顺序不当。

3、死锁产生的必要条件

① 互斥条件

一段时间内某种资源仅能被一个进程占用。

② 不剥夺条件

进程所获得的资源在使用完毕之前，不会被其他进程强行夺走。

③ 请求与保持条件（部分分配条件）

进程每次申请它所需的一部分资源，该条件也称部分分配条件。

④ 环路等待条件

存在一种进程资源的循环等待链，而链中的每一个进程已经获得的资源同时被链中的下一个进程请求。

预防死锁的方法

要想防止死锁的发生，只需破坏死锁产生的4个必要条件之一即可。

存在一个安全序列<P1,P2,...,Pn>，按此分配资源，可使每个进程顺利完成。

可将系统状态分为安全状态和不安全状态（可能发生死锁），只要能使系统处于安全状态，便可以避免死锁的发生。

破坏就是安全状态。 避免死锁的实质：系统在进行资源分配时，如何使系统不进入不安全状态。/让系统处于安全状态

4、考 **银行家算法**

MANA

数据结构

设有n个进程，m种系统资源。

available[m]：当前m种资源的剩余数量

max[n,m]：每个进程对每种资源的最大需求。

need[n,m]：每个进程还需要的资源量。

allocation[n,m]：每个进程现在已有的资源的数量。

need[i,j] = max[i,j] - allocation[i,j]

资源请求 4列

	Max	Allocation	Need	Available
P_0				
P_1				
P_2				

安全性算法 5列 WNA WA F

Work Need Allocation Work+Allo Finish

安全性算法（确定计算机是否处于安全状态）

系统可提供给进程各类资源的数量

① 令work[m] = available[m]

finish[i] = false, i=0,...,n-1

② ~~for(int i=0; i<n; i++)~~ 在集合中找到1个i使满足下个if，即③，若找不到转④

③ if(finish[i]==false && $need_i$ ≤ work){

work=work+$allocation_i$

finish[i]=true;

}

④ if finish[i]都==true, 则安全

资源请求算法（判断是否可安全允许请求）

设$request_i$为进程P_i的请求向量(长度为m)

① if($request_i$ ≤ $need_i$){

② if($request_i$ ≤ available){

③ Available = available - $request_i$;

$allocation_i$ = $allocation_i$ + $request_i$;

$need_i$ = $need_i$ - $request_i$

④ if(safe) return;

else 恢复→逆③

}

else{等待}

}

else{

错误出错；

}

1、处理器的三级调度（名称英文、任务） 考

2、常见调度算法 考

“[illegible]作删除部分”

3、死锁的概念、原因（2个）、~~特征（4个）~~、必要条件（4个） 考

4、银行家算法（资源请求+安全性算法） 考

5、调度算法评价标准 考