**基本概念回顾**

# [1死锁的定义：](#_1死锁的定义：)

# [2死锁产生的原因：](#_2死锁产生的原因：)

# [3死锁的四个必要条件：](#_3死锁的四个必要条件：)

# [4资源分配图及其化简方法：](#_4资源分配图及其化简方法：)

# [5死锁定理：](#_5死锁定理：)

# [6死锁恢复的方法：](#_6死锁恢复的方法：)

# [7死锁预防的方法：](#_7死锁预防的方法：)

# [8系统安全/不安全状态：](#_8系统安全/不安全状态：)

# [9银行家算法：](#_9银行家算法：)

# 1死锁的定义：

　　死锁是计算机系统中多道程序并发执行时，两个或两个以上的进程由于竞争系统资源，而出现的一种互相等待的现象。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 2死锁产生的原因：

    一是系统竞争临界资源。当系统所拥有的某种临界资源个数比各个进程要求该资源的总数要少时，则有可能引起多个并发进程对资源的竞争而产生死锁；二是进程推进顺序不当．进程运行过程中，由于请求和释放资源的顺序不当，而产生死锁．

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 3死锁的四个必要条件：

　（1）互斥条件 ：相应的资源是独享的，在一段时间内某资源只能由一个进程占有。

　（2）不可剥夺条件：进程在运行中所获得的资源未使用完之前，不能被其它进程"抢占"。

　（3）部分分配条件：进程在整个运行期间所需要的全部资源并不能一次分配完毕，每次只能申请所需要的部分资源。

　（4）环路等待条件：死锁发生时，一定有一种进程和资源的循环链。其中每个进程已获得的资源同时被链中下一个进程所请求。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 4资源分配图及其化简方法：

　　所谓资源分配图即是用来描述进程、资源及它们之间关系的一种有向图。

  假设某个资源分配图中存在一个进程Pi，此刻Pi是非封锁进程，那么我们可以对此资源分配图做以下化简：当Pi有请求边时，首先将其请求边变成分配边，即满足Pi进程的相应资源请求。而一旦Pi进程的所有资源请求都得到满足，Pi就能在有限时间内运行结束，并释放它所占用的所有资源。所以此时Pi只有分配边，我们可以将所有这些分配边删去。总之，对非封锁进程Pi的化简即删除资源分配图中与Pi连结的所有有向边，使Pi变成孤立结点。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 5死锁定理：

  系统中某个时刻S为死锁状态的充分必要条件是，S时刻系统的资源分配图是非完全可化简的。利用死锁定理，我们可以通过某种算法来实现资源分配图的化简，并以此为依据检测系统中是否存在死锁以及哪些进程是死锁的。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 6死锁恢复的方法：

  一种是强制性地从系统中撤消一些死锁进程，并剥夺它们的资源给其余的进程；另一种是使用一个有效的挂起和解除机构，来挂起一些进程，以便从被挂起进程中剥夺一些资源用来解除死锁。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 7死锁预防的方法：

　　（1）打破"不剥夺条件：强迫那些请求新资源而没有立即得到满足的进程，释放它已保持的其它资源。即一个进程已占用的资源在运行过程中可能要暂时释放。

　　（2）打破"部分分配"条件：对某进程所要求的资源一次性地分配完毕。这样，进程在运行过程中就不再需要新的资源。这种方法又称为预先静态分配法。但在做静态分配时，只要有一种资源不能满足，该进程就必需等待．

　　（3）打破"环路等待"条件：在资源的分配过程中，对资源的请求作出某种限制，使环路不可能出现。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 8系统安全/不安全状态：

　　（1）安全状态：所谓安全状态，是指系统能按照某种顺序为每个进程分配所需的资源，使得每个进程都能顺利完成。

　　（2）不安全状态：所谓不安全状态，即某个时刻系统中不存在一个安全序列，能使所有的进程都顺利完成，这时我们称系统处于不安全状态．

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)

# 9银行家算法：

　　当第i个进程需要请求系统资源时，我们就将其对各个资源的需求量送入Re数组中，然后系统依次执行。

　　(1)若j从1到M，所有Re(j)＜=Need(i，j)，则程序可继续，否则说明i进程的这次请求数量，超出了事先给定的它对Rj资源的最大需求，转出错处理。

　　(2)若j从1到M，所有Re(j)＜=Av(j)，则程序可继续，否则代表系统中尚无足够的资源满足这次请求，Pi必须等待。

　　(3)满足Pi这次请求，j从1到M顺序执行：

      (4)执行安全性算法：通过调用安全性算法，检查这次资源分配后，系统是否处于安全状态．若安全状态标志为真，说明系统是安全状态，这次分配可能进行，银行家算法运行结束。否则，要恢复刚才分配前的各资源状态，同时让Pi进程等待。

[http://cs.hytc.edu.cn/lab/cl/%B2%D9%D7%F7%CF%B5%CD%B3/image/zpjs0600004.gif](#_1进程的定义:_1)