操作系统综合题知识点笔记

可能会算特别大的数字，请带上计算器。

1. 信号量（semaphore）

对一个信号量有 2 种操作：down&up（或者用 P&V 来表示）。

down 操作检查信号量 S 是否大于 0，是则将 S 减 1 并继续，否则使进程睡眠。up 操作检查有无进程在 S 上睡眠，是则唤醒其中一个进程，否则将 S 加 1。

注意，旧版 down 操作无论 S 是否大于 0 都将 S 减 1；旧版 up 操作无论有无进程在睡眠都将 S 加 1。即 down 操作检查信号量 S 是否大于 0，是则将 S 减 1 然后继续，否则将 S 减 1 然后使进程睡眠。up 操作检查有无进程在 S 上睡眠，是则将 S 加 1 然后唤醒其中一个进程，否则将 S 加 1。

题目默认隐含的假设：所有进程都在开始时就启动了。不能在中途启动进程。

光看这个没什么用，做做课后题或往年试题。

2. 调度（scheduling）

周转时间（turnaround time）：一个批处理作业从提交到完成的统计平均时间。

忽略进程切换（process switch，又称上下文切换 context switch）的时间，一个进程从提交到完成的时间等于等待时间与运行时间之和。原因：一个进程从提交到完成要么在等待，要么在运行。（答题时最好抄一下原因，证明一下再用）

答题时请列出进程运行的时间轴（推荐用甘特图（Gantt chart）代替，比较繁琐但是思路清晰）。

A. 先来先服务（First-Come First-Served, FCFS）：顾名思义。有护航效果（Convoy Effect）。

B. 最短作业优先（Shortest Job First, SJF）：顾名思义。

C. 最短剩余时间优先（Shortest Remaining Time Next）：顾名思义。与最短作业优先的区别：假如一个长作业（进程）正在运行时，有一个短作业（进程）到达，SJF 不会将正在运行的长作业中断然后让短作业先运行，而最短剩余时间优先会。

D. 转轮调度（Round Robin）：先设定时间片（quantum）的值，让进程按照先来后到排队，将队首的进程运行一个时间片后，把它放到队尾，然后运行下一个在队首的进程。

E. 优先级调度（Priority Scheduling）：顾名思义。低优先级进程可能会有饥饿（starvation）现象。

F. Multi-Queue Scheduling：设定好每一级队列占用 CPU 时间的比例，每个优先级都有一个队列，把不同优先级的进程放入到不同队列中。

G. 多级队列（Multi-Level Feedback Queue）：一个运行时间很长的进程刚开始被放在最高优先级的队列，运行 1 个时间片，然后被放入次高优先级的队列中，运行 2 个时间片……最后到达最低优先级的队列，或者在中途就运行完了。在最低优先级的队列中用 FCFS 或者其他算法调度。

H. 最短进程优先（Shortest Process First，SPF）：顾名思义。

I. 保证调度（Guaranteed Scheduling）：向用户做出明确的性能保证，然后去实现它。计算各个进程应得的 CPU 时间，跟踪每个进程，算出实际获得的时间与应得的时间之比，然后转向最低的进程，直到该进程的比率超过最接近的进程。

J. 彩票调度（Lottery Scheduling）：将一些彩票分配给不同进程，每个进程获得的数量由优先级决定，然后随机抽出一张，拥有这张彩票的进程获得一定的资源。

K. 公平分享调度（Fair-Share Scheduling）：考虑每个用户，让使用 CPU 时间少于应得份额的用户优先运行进程。

3. 虚拟内存（Virtual Memory）

分页（paging）：程序不直接通过物理地址访问内存，而是通过虚拟地址（virtual address）访问。所有虚拟地址构成虚拟地址空间（virtual address space）。内存管理单元（Memory Management Unit, MMU）负责将虚拟地址映射为物理地址。虚拟地址空间按照固定大小划分成页面（page）的单元，在物理内存中被称为页框（page frame）。它们的大小通常一样。

缺页错误（page fault）：一个页面不再内存而只在磁盘中。

实质就是将内存作为磁盘的缓存使用。

一个虚拟地址分为虚拟页号和偏移量（offset）。前者指向相应的页面，后者表示页面中的字节偏移（类似于数组下标）。在决定虚拟地址位数时，要令每一个字节都有一个虚拟地址与其一一映射。

虚拟地址转换成物理地址：虚拟页号转换成页框号（物理页号），然后与偏移量拼接在一起。

多级页表：以两级为例，将虚拟地址分为 PT1、PT2 和偏移量三部分，分级查找。

倒排页表（inverted page table）：建立一张用页框号映射虚拟页号的表。转换检测缓冲区（Translation Lookaside Buffer, TLB）找不到相应页表时，在倒排页表中遍历对应的虚拟页号。但通常虚拟页号会使用散列表索引，即建立一个槽数等于页框数的散列表，虚拟页号为键，页框号为值，页表项即键值对，将页表项放在散列表中。

倒排页表地址转换（物理到虚拟）：算出页框号，找到倒排页表（下标从 0 开始）该下标的对应项。把对应项的虚拟页号与偏移量拼接。

4. 页面置换算法

A. 最优页面置换算法（OPT）：发生缺页时，有些页面在内存中，其中有一页将很快被访问（也包含紧接着的下一条指令的那页），而其他页面则可能要到 10、100 或者 1000 条指令后才会被访问，每个页面都可以用在该页面首次被访问前所要执行的指令数进行标记。标记最大的页应该被置换。

这个算法唯一的一个问题就是它无法实现。当缺页发生时，操作系统无法知道各个页面下一次是在什么时候被访问。

B. 最近未使用页面置换算法（Not Recently Used, NRU）：在每个页面上设置 R 位，一开始为 0，如果被访问了就置为 1。定期将所有 R 位置为 0。（实际上可以只考虑内存中页面的 R 位）当需要淘汰页面时，随机淘汰掉 R 为 0 的页面。（随机淘汰两次 R 位清零之间未被访问的）

C. 先进先出页面置换算法（First-In First-Out, FIFO）：顾名思义。

D. 第二次机会（second chance）页面置换算法：将被调入内存的页面按照先后顺序放入队列中，为每个页面设置 R 位。当要置换页面时，取出队首的页面，如果 R 为 0 就淘汰掉，否则将 R 置为 0 并放到队尾，再取出下一个在队首的页面执行相同操作，直到有页面被淘汰为止。

E. 时钟（clock）页面置换算法：与第二次机会页面置换算法基本相同，只是队列换成环形链表，用一个指针指向最老的页表，如果 R 为 0 就淘汰，否则将 R 置为 0，指针指向下一个，重复，直到淘汰一个页面为止。

F. 最近最少使用页面置换算法（Least Recently Used, LRU）：缺页错误时，置换未使用时间最长的页面，即最后一次被访问时间最前的页面。（只考虑内存中页面的访问时间也行）

G. 最不常用算法（Not Frequently Used, NFU）：统计每个页面被访问的次数，淘汰掉次数最小的。

H. 老化（aging）算法：为每个页面保存一个变量（计数器），每次时钟滴答时将该变量右移一位，如果在时钟滴答间访问了一个页面，就将变量最高位置为 1。要淘汰页面时，就淘汰变量最小的页面。

I. 工作集（working set）算法：一个进程当前正在使用的页面集合称为它的工作集。每个页面有一个上次使用时间的变量和 R 位。假定每个时钟滴答都会清除 R 位。当缺页中断发生时，检查所有页面，若为 1，则将上次使用时间置为当前时间，否则，检查生存时间（当前时间减上次使用时间之差），大于 τ 则淘汰该页面并继续扫描其他页面，否则就将其记录下来。全部扫描完之后，如果没有页面被淘汰，就淘汰生存时间最长的，如果 R 全为 0 就随机淘汰一个。

J. 工作集时钟算法（WSClock）：与时钟算法类似，当 R=1 时 R 置零，上次使用时间不变，当 R=0 时淘汰生存时间大于 τ 的页面，否则记录下来。如果一圈之后没有页面被淘汰，置换记录中生存时间最大的，如果没有被记录的就再扫一次？？？

5. i 节点（i-Node）

（文件的）i 节点=文件属性（可能没有）+磁盘块 0 的地址（一级指针）+磁盘块 1 的地址+...+磁盘块 n 的地址+指针块的地址（N 级指针）

给定文件路径写磁盘读取次数：每个目录两次（含根目录，目录的 i 节点一次，目录项一次），最后的文件一次（i 节点）。

设（有效大小）文件的最大大小为 M，b 为一个磁盘块的大小，l 为磁盘指针长度，第 i 级指针数量为c_i，则 $M=\sum_{i=1}^{n} c_i[\frac{b}{l}]^{i-1}$ 。

6. 磁盘臂调度算法

答题时列出读取顺序。

A. 先来先服务（First-Come First-Served, FCFS）：顾名思义。

B. 最短寻道优先（Shortest Seek First, SSF）：总是处理于磁头最近的请求。

C. 电梯算法（elevator algorithm）：先处理当前方向上面所有的请求，处理完之后磁头反向，再处理反方向所有的请求。设请求磁盘地址最大值为 M，最小值为 m，初始地址为 i，磁头运行路程为 s。若初始方向为从小往大，则 s=(M-m)+(M-i)，否则 s=(M-m)+(i-m)。

7. 银行家算法（banker's algorithm）

资源分配图：进程 A 占有资源 R：R→A。进程 B 请求资源 S：B→S。成环则死锁。

安全状态：没有死锁发生，并且即使所有进程突然请求对资源的最大需求，也存在某种调度次序能够使得每一个进程运行完毕，则称该状态是安全的，否则称为不安全状态。

银行家算法：如果满足请求之后还在安全状态，则满足之，否则，推迟（拒绝）之。

检测算法：设有进程 1 到 n，现有资源向量为 E，已分配资源之和为 $P=\sum_{i=1}^{n} C_i$，可用资源为 $A=E-P$，进程 i 的分配向量为C_i，最大请求向量为R_i，令仍然需要的资源

$S_i = R_i - C_i$。定义向量 X>Y 为向量 X 每一项都大于 Y 的对应项。

（1）遍历 $S_i, i = 1,2,...,n$，检查是否有 $S_i \le A$ 且进程 i 未被标记终止。

（2）若有，则将 i 进程标记为终止并令 $A = A + C_i$。

（3）如果有进程满足条件，重复以上两步，否则，检查所有进程是否被标记，是则初始状态为安全的，否则可以确定发生了死锁。