进程调度问题.

首先注意以下问题.

1. 优先级怎么样是高优先级. 2. 同时到达如何处理 3. 在某些方法中能否直接处理.

a). FCFS.

只需考虑到达时间, 将到达时间在等达对列中排序, 然后按序处理.

b). SJF.

只考虑虎剩余时间, 开始一个新的进程前, 对比当前等待队列中.
谁的运行时间短, 就运行谁. 在运行完成前, 不做考虑, 不被打断.

c). SRTN

与SJF相同. 不同的是每一步都需对比一下, 当前队伍中是否有
运行时间更短的进程. 也就是当有新进程到达时, 会可能有变化.

d). Round Robin mit x

对一个进程只运行x秒. 如果这个进程完成, 则运行队列中的
下一个进程. 如果未完成, 则将这个进程放到队列最后. 运行队列
中的下一个进程. 也就是每x秒, 都考虑一下等待队列.

e). nicht - priemptive Priority

只考虑优先级. 运行完一个进程后, 对比当前队列中的优先级. 然后处理.
注意: 当队列中有优先级更高的进程加入, 只会在等待队列中更优先, 不抢占.
则入队的进程需在之后进入运行. 加入时间和运行不能同一时间.

f). priemptive Priority.

与 nicht 相同. 不同的是每一步都需对比一下. 当前队伍中是否有
优先级更高的(但注意, 刚加入的进程在t后运行. 例如 在t=4
时到达一个优先级更高的, 那么只能t=5运行. 如果 t=4时要换下一个
进程, 则运行原等列中优先级更高的) × 注意: 第一个是0

g). EDF

和 Priemptive Priority 一样. 只是优先级换为一个新参量 Deadline.
一个进程的优先级是固定不变的. 但 Deadline 会根据时间减小.
等每一步都算一下, 队列中各个进程的 Deadline.

Durchschnitt Wartezeit: 首先找到何时加入等待队列, 再找到加入等待队列
的时间. 做减法, 求平均数. (也就是在等待队列的总时间), 可以
算一下. 从到达时间算起, 到最后一个这个进程运行. 中间有多久
没有运行它.

Durchschnitte Verweildauer: 首先找到何时加入队列, 再找到运行的最后一秒.
做减法. 求平均数. 可以数一下. 从到达的那一秒算起, 到运行完成
总共经历多少秒.

银行家算法:

会给出三个列表: 分别是已分配资源 C, 最大总资源 M, 仍需资源 R

1. 如果总给其中两个, 则计算. C + R = M
2. 计算初始拥有资源: 总资源 E = 可用资源 + Σ C (已给资源之和): A
3. 对比当前可用资源和各个进程的 R, 当 A 满足 R
则运行这个 Process.
4. 这个新的可用资源为之前可用资源 An-1 + 已分配资源 Cp = An
5. 如果所有进程都完成, 则 Sicher. 如果某一步资源不够, 则 Unsicher

伙伴系统: "见缝插针"

4KB 8KB 16KB 32KB 64KB 128KB 256KB

装箱问题: 按顺序放入数据

1. First Fit: 在所有的空位置中选第一个可以放进去的
2. Best Fit: 在所有的空位置中选冗余最小的
3. Worst Fit: 在所有的空位置中选冗余最大的.
4. Next Fit: 从上一个放入的位置开始算. 找到下一个合适的位置.

Paging:

1. FIFO: 每次更换时, 看前一列谁连续的最长, 最长的就是
最先入队的
2. LRU: 每次更换时. 看前一列在Frame中, 谁离的最远, 因为有
重复的现象, 所以最长的不一定最近.
3. Second - Hand:
4. Belodys Optimalalgorthimus

求分配内存.

顺序从左向右. 第一位是逻辑为1.

写入: 占几个块, 就加几个1, 并且这几个位置属于这个Datei

删除: 把这个Datei对应的位置写回0

占位: 则跳过. 写下一个为0的位置.

FAT链.

1. 链的最后一个是EOF
2. 除EOF之外, 每一个位置填的是链的下一个数.
例 Datei: 5 - 9 - 12, 则 5 填 9. 9 填 12. 12 填 EOF.
3. 没有被用到的写 FREE.

Dijkstra算法

1. 初始时, 集合S包含起点, 与起点直接连接的点可以求距离.
2. 选择目前已知距离的点中, 距离最小的, 加入集合.
3. 更新到各点的距离. 主要是与新加入的点有关联的点, 看距离会不会减小
具体为将原距离与新加入的距离 + 这个点到新加入的距离)
4. 最短路径: 这个位置开始, 找各个相关点, 连为一条线

TCP-Protokoll

1. 图中, 横轴为时间点 Zeitpunkt, 纵轴为拥塞窗口 Congestion Window
2. 当拥塞窗口小于阈值 threshold, 会慢启动, 指数增长.
3. 当拥塞窗口大于阈值 threshold, 会进入拥塞避免算法.
4. 当收到 dreier doppelter 时, 阈值设为当前窗口的一半. 从这个点开始慢启动
5. 当收到 Time Out 时, 阈值设为当前窗口的一半. 然后窗口为1开始慢启动
6. 若检查某个点的阈值. 则看它前一个阶段的终止点, 阈值为终止点的一半.
7. 若检查某个段在某个时间点, 则将前面的值累加

Adresse berechnen

1. 给一个IP地址的子网 Subnetz 和子网掩码 Netzmaske