

2019操作系统试卷回忆

印象很不深刻，期待大家补充。

一、选择题

基本是MOOC原题，但稍有变化，请注意。

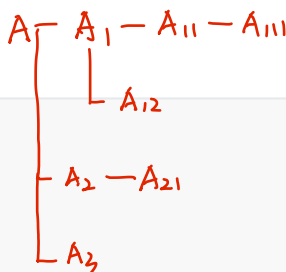
二、简答题

今年没有考银行家算法，非常奇怪。

1. 自顶向下简述I/O软件的四个层次。
2. (电梯调度算法, 只是换了个数据)
3. `fork()`的进程数, 并画出进程树。

用户空间的 I/O 软件、独立于设备层的 I/O 软件、设备驱动程序、中断处理程序

```
main(){
    fork(); // pc, 进程A
    fork();
    fork();
}
```



4. 12个直接地址，每个盘块1024B，每个盘块可以存放256个盘块地址，计算50M的文件和100M的文件各自占用的盘块数。

$$(50 \times 1024 \text{KB} - 12 \text{KB} - 256 \text{KB}) / 1024 \text{B} = 50932$$

 12直. 256 - 50932 =
5. 简述虚存分页的原理，并画出流程图。

$$(50 \times 1024 \text{KB} - 12 \text{KB} - 256 \text{KB}) / 1024 \text{B} = 50932$$

12 直. 256 - 50932 =

12直. 256- . 50932 =

三、计算题

页架：页表，页号

页架：页表

页号偏移

写出执行序列：

页架

页表

页架

$$(100 \times 1024 \text{ KB} - 12 \text{ KB} - 256 \text{ KB} - 44 \times 1024 \text{ KB}) / 1024 = 36596$$

1. 进程调度, 写出执行序列:

进程	到达时间	处理时间
A	0	1
B	0.5	9
C	1.5	1
D	2.5	9

1. $RR_{q=1}$

A B C D D D D D D D D D D

2. FBack,q=1

ABC D BD BD BD BD BD BD BD BD BD BD

3. FBack, $q=2^i$

ABCD BBDD BBBB DDDD BDD

- ## 2. 页面替换算法：LRU和Clock

- ### 3. (作业原题)

有一多道程序设计系统，1) 进程调度（处理器低级调度）采用时间片调度算法，不考虑进程的输入输出和操作系统的调度开销；2) 存储管理采用可变分区方式，用户空间为100K，采用最先适应算法分配主存且不允许移动；3) 系统配有4台磁带机，对磁带机采用静态分配策略。今有如下作业序列：

作业名	进输入并时间	需执行时间	主存量要求	申请磁带机数
J1	10:00	25分钟	15K	2 11:10~12:00
J2	10:20	30分钟	60K	1 12:00~12:40
J3	10:30	10分钟	50K	3 11:00~11:10
J4	10:35	20分钟	10K	2 11:40~12:20
J5	10:40	15分钟	30K	2 11:10~11:40

假定操作系统从11:00开始调度，当作业调度采用“响应比最高优先算法”时，分别写出J1、J2、J3、J4、J5装入主存和离开主存的时间。

11:00 J₃进 0~50 ③
J2装入主存时间：； J3装入主存时间：11:10 J₃完 J₁进 J₅进 0~15 16~45 ④

11:40 J₅完 J₄进 0~15 16~25 ④
J4装入主存时间：； J5装入主存时间：12:00 J₁完 J₂进 16~25 26~85 ⑤

4. （感谢tcj提供）PV操作。

一个仓库，最多能放A，B产品各m个。每次生产需要A，B产品各一个。两组供应商分别生产A，B产品。当某个产品的数量比另一个产品数量多n(n<m)个时，仓库暂时停止该产品的进货，集中进货另一个产品。

5. （感谢tcj提供）管程。

三个生产者P1，P2，P3进行生产活动，需要从供应商仓库获得原料。原料有三种：糖，橘子精，水。每个生产者已经有其中的两种。当容器为空时，供应商会往容器放一种原料。P1已经有：糖，橘子精 P2已经有：糖，水 P3已经有：橘子精，水

semaphore mutex=1, int a=0, b=0

```

void consumer() {
    while(1) {
        p(mutex)
        if (a > 0 && b > 0) {
            // 得到原料
            a--;
            b--;
            v(mutex)
            // 生产
        } else v(mutex)
    }
}

void producer() {
    while(1) {
        p(mutex)
        if (a-b > n) b++
        else if (b-a > n) a++
        else if (a < m && b == m) a++
        else if (a == m && b < m) b++
        else if (a < m && b < m) (a或b随机一个)++
        v(mutex)
    }
}

```

Semaphore G; int G-count
type company = monitor
Interface Module IM
Semaphore S[i] int S-count[i]
define get, put int plate = -1
use enter, leave, wait, signal
void get(int i) {
 enter(IM)
 if (plate != i)
 wait(S[i], S-count[i], IM)
 signal(G, count, IM) leave(IM)
}
void put(int i) {
 enter(IM)
 if (plate != -1)
 wait(G, count, IM)
 plate = i
 signal(S[i], S-count[i], IM)
 leave(IM)
}