

更正：

- 题2，误将Max0112看成0012，已修正
- 题3，误将每块4KB写成1KB，已修正
- 题4，独木桥问题，已修正
- 题7，处理一次缺页时间是100ms，不是100ns，已修正

感谢网友 [ve1pro](#) 和 [Jackson](#) 的指正。🙏

考察知识点：进程同步

1. 系统中有三个进程GET、PRO和PUT，共用两个缓冲区BUF1和BUF2。假设BUF1中最多可放11个信息，现已放入了两个信息；BUF2最多可放5个信息。GET进程负责不断地将输入信息送入BUF1中，PRO进程负责从BUF1中取出信息进行处理，并将处理结果送到BUF2中，PUT进程负责从BUF2中读取结果并输出。试写出正确实现GET、PRO、PUT的同步与互斥的算法。（要求：用类C语言伪代码描述，条理清楚，注释得当；信号量原语统一使用P和V）

参考解答：

定义如下信号量

empty1=9，表示BUF1剩余可放信息的个数，BUF1最多可放11个信息，现已放入了2个信息，则其剩余可放信息的个数为9。

empty2=5，表示BUF2剩余可放信息的个数，BUF2最多可放5个信息，现已放入了0个信息，则其剩余可放信息的个数为5。

full1=2，表示BUF1已经存放的信息个数，BUF1已经放入了2个信息。

full2=0，表示BUF2已经存放的信息个数，BUF2已经放入了0个信息。

mutex1=1，表示锁信号量，用来使进程GET和PRO互斥地使用CPU。

mutex2=1，表示锁信号量，用来使进程PRO和PUT互斥地使用CPU。

```
GET(){
    while(1){
        P(empty1);
        P(mutex1);
        输入信息送入BUF1中
        V(mutex1);
        V(full1);
    }
}

PRO(){
    while(1){
        P(full1);
        P(mutex1);
        从BUF1中取出信息处理
        V(mutex1);
        V(empty1);

        P(empty2);
        P(mutex2);
        将处理结果送入到BUF2中
        V(mutex2);
        V(full2);
    }
}
```

```

    }
}

PUT(){
    while(1){
        P(full2);
        P(mutex2);
        从BUF2中读取结果并输出
        V(mutex2);
        V(empty2);
    }
}

void main(){
    cobegin
        GET();
        PRO();
        PUT();
    coend
}

```

考察知识点：系统安全状态、银行家算法

2. 某时刻系统的A、B、C、D四种资源状态如下表所示：

Process	Allocation	Max	Available
P0	0012	0112	1540
P1	1000	1750	
P2	1354	2356	
P3	0014	0656	

- (1) 系统中四类资源各自的总数是多少？
- (2) 请写出Need矩阵。
- (3) 当前系统状态是否安全？请写出一个安全序列。
- (4) 如果P1发出请求(0, 2, 2, 0)，是否可以满足该请求？如果可以，请给出安全序列。

参考解答：

Allocation：已分配给该进程的资源数

MAX：进程对资源的最大需求数

Available：可获得的资源数

(1) 将Allocation和Available中的各类对应的资源数相加

A类资源的总数=1+1+1=3

B类资源的总数=5+3=8

C类资源的总数=1+4+5+1=11

D类资源的总数=2+4+4=10

故系统中四类资源各自的总数是3、8、11、10

(2)

Need=Max-Allocation

故Need的内容如下图所示

Process	Need
P0	0100
P1	0750
P2	1002
P3	0642

Need矩阵表示：

$$\begin{bmatrix} P0 & 0100 \\ P1 & 0750 \\ P2 & 1002 \\ P3 & 0642 \end{bmatrix}$$

经上述步骤后，完整的资源分配情况如下图所示：

Process	Allocation	Max	Need	Available
P0	0012	0112	0100	1540
P1	1000	1750	0750	
P2	1354	2356	1002	
P3	0014	0656	0642	

(3)
 利用安全性算法对此刻的资源分配情况进行分析，可得到如表所示的安全检测情况。
 从表中可以看出，此时此刻存在着一个安全序列{P0,P2,P1,P3}，故该系统是安全的。

Process	Work	Allocation	Need	Work+Allocation	Finsh
P0	1540	0012	0100	1552	True
P2	1552	1354	1002	2 8 10 6	True
P1	2 8 10 6	1000	0750	3 8 10 6	True
P3	3 8 10 6	0014	0642	3 8 11 10	True

(4)

P1发出请求Request(0,2,2,0)，按照银行家算法检查：

Request(0,2,2,0)<Need(0,7,5,0)

Request(0,2,2,0)<Available(1,5,4,0)

试分配并修改相应数据结构，由此形成的资源分配情况如图所示：

Process	Allocation	Max	Need	Available
P0	0012	0112	0100	1320
P1	1220	1750	0530	
P2	1354	2356	1002	
P3	0014	0656	0642	

利用安全性算法对此刻的资源分配情况进行分析，可得到如表所示的安全检测情况。

从表中可以看出，此时此刻存在着一个安全序列{P0,P2,P1,P3}，故该系统是安全的。

故P1发出请求(0,2,2,0)，能满足该要求，安全序列为{P0,P2,P1,P3}。

Process	Work	Allocation	Need	Work+Allocation	Finish
P0	1320	0012	0100	1332	True
P2	1332	1354	1002	2686	True
P1	2686	1220	0530	3 8 10 6	True
P3	3 8 10 6	0014	0642	3 8 11 10	True

考察知识点：逻辑地址到物理地址的转换

7. 系统内存被划分成8块，每块4kB。某作业的虚拟地址空间共划分成16个页面。当前在内存的页与内存块的对应关系如下表所示，未列出的页表表示不在内存。

页号	块号	页号	块号
0	2	4	4
1	1	5	3
2	6	9	5
3	0	11	7

试指出对应于下列虚拟地址的绝对地址：

(1)20B

(2)4100B

(3)8300B

参考解答：

虚拟地址-->物理地址的转换规则：

偏移量保持不变，页号转换成物理块号

页号=虚拟地址/页面大小并向下取整

偏移量=虚拟地址 MOD 页面大小

页面大小4KB=4096B

(1)

页号=20/4096=0

页内偏移量=20 MOD 4096=20

查页表得，页号为0，对应的块号为2，

由于每块的长度为4KB，所以第2块的起始地址为8192【块号从0开始编号】

∴虚拟地址20对应的绝对地址是8192+20=8212

(2)

页号=4100/4096=1

页内偏移量=4100 MOD 4096=4

查页表得，页号为1，对应的块号为1，

由于每块的长度为4KB，所以第1块的起始地址为4096【块号从0开始编号】

∴虚拟地址4100对应的绝对地址是4096+4=4100

(3)

页号=8300/4096=2

页内偏移量=8300 MOD 4096=108

查页表得，页号为2，对应的块号为6，

由于每块的长度为4KB，所以第6块的起始地址为24576【块号从0开始编号】

∴虚拟地址8300对应的绝对地址是24576+108=24684

考察知识点：进程同步

2、(8分) 用信号量解决“独木桥”问题：同一个方向行人可连续过桥，当某一方向有人过桥时，另一个方向的行人必须等待；当某一方向无人过桥时，另外方向的行人可以过桥。

(1) 本问题中有哪些同步或互斥关系？

(2) 给出两个方向任一行人通过该独木桥的同步算法。

参考解答：

(1)

同步关系：同一方向的行人之间

互斥关系：不同方向的行人之间

(2)

定义如下信号量：

将独木桥的两个方向标记为A，B，

countA=0，表示从A方向往B方向过桥的人数，初值为0，表示一开始，从A方向往B方向过桥的人数为0；

countB=0，表示从B方向往A方向过桥的人数，初值为0，表示一开始，从B方向往A方向过桥的人数为0；

mutexA=1，表示锁信号量，用来控制对countA的操作；

mutexB=1，表示锁信号量，用来控制对countB的操作；

mutex=1，表示锁信号量，用来实现两个方向的行人对独木桥的互斥使用。

```
ADirect(){
    P(mutexA);
    if(countA==0){
        P(mutex);
    }
    countA=countA+1;
    V(mutexA);
    通过独木桥;
    P(mutexA);
    countA=countA-1;
```

```

        if(countA==0){
            V(mutex);
        }
        V(mutexA);
    }

    BDirect(){
        P(mutexB);
        if(countB==0){
            P(mutex);
        }
        countB=countB+1;
        V(mutexB);
        通过独木桥;
        P(mutexB);
        countB=countB-1;
        if(countB==0){
            V(mutex);
        }
        V(mutexB);
    }

    void main(){
        cobegin
            ADirect();
            BDirect();
        coend
    }

```

考察知识点：多级间址

3、(8分) 在某个采用混合索引分配的文件系统中，FCB中有 $i_addr[0] \sim i_addr[8]$ 共9个物理地址项，其中 $i_addr[0] \sim i_addr[6]$ 是7个直接地址项， $i_addr[7]$ 是1个一次间址项， $i_addr[8]$ 是1个二次间址项。如果一个盘块的大小是4KB，每个盘块号占4个字节。请写出将下列文件的字节偏移量转换成物理地址的过程：

(1) 10000; (2) 500000。

参考解答：

(1)

$\because 1KB=1024B$

一个盘块的大小为4KB

\therefore 一个盘块的大小为4096B

$10000/4096=2 \dots 1808$

即从FCB的第2个地址项- $i_addr[1]$ 获得物理盘块号，块内偏移量为1808

(2)

$500000/4096=122 \dots 288$,

每个盘块最多存放的盘块号数量=一个盘块大小/每个盘块号大小=4KB/4=1024

即每个盘块最多存放的盘块号数量为1024。

第122个地址项属于一次间址项，

$122-7=115$ ，减去7个直接间址项，得其在一次间址项中的第115个地址。

\therefore 从 $i_addr[7]$ 的第115个地址获得物理盘块号，块内偏移量为288

考察知识点：系统安全状态、银行家算法

4、(8分) 在银行家算法中，若出现下述资源分配情况：

Process	Allocation	Need	Available
P0	0032	0012	1622
P1	1000	1750	
P2	1354	2356	
P3	0332	0652	
P4	0014	0656	

试问：(1) 该状态是否安全？（必须写出安全性检查的过程）

(2) 若进程 P2 提出请求 Request (1, 2, 2, 2) 后，系统能否将资源分配给它？为什么？

参考解答：

(1) 利用安全性算法对此刻资源分配情况进行分析，可得到如表所示的安全检测情况。

从表中可以看出，此时此刻存在着一个安全序列{P0, P3, P1, P2, P4}，故该系统是安全的。

Process	Work	Allocation	Need	Work+Allocation	Finish
P0	1622	0032	0012	1654	True
P3	1654	0332	0652	1986	True
P1	1986	1000	1750	2986	True
P2	2986	1354	2356	3 12 13 10	True
P4	3 12 13 10	0014	0656	3 12 14 14	True

(2)

P2提出请求Request(1,2,2,2)后，按照银行家算法检查：

Request(1,2,2,2)<Need(2,3,5,6)

Request(1,2,2,2)<Available(1,6,2,2)

试分配并修改相应数据结构，由此形成的资源分配情况如图所示：

进行安全性检查，可用资源Available(0,4,0,0)已不能满足任何进程的需要，故系统进入不安全状态，此时系统不分配资源。

Process	Allocation	Need	Available
P0	0032	0012	0400
P1	1000	1750	
P2	2576	1134	
P3	0332	0652	
P4	0014	0656	

考察知识点：请求分页管理系统

5、(8分) 某请求分页管理系统，假设进程的页表如下：

页号	页框号	有效位	装入时间
0	101H	1	2
1	—	0	—
2	254H	1	4

页面大小为 4KB，一次内存的访问时间为 100 纳秒 (ns)，一次快表 (TLB) 的访问时间是 10ns，处理一次缺页的平均时间为 100 毫秒 (已含更新 TLB 和页表的时间)，进程的驻留集大小固定为 2 个页框，采用 FIFO 法置换页面。假设 1) TLB 初始为空；2) 地址转换时，先访问 TLB，若 TLB 未命中时再访问页表 (忽略 TLB 更新时间)；3) 有效位为 0 表示页面不在内存中。

请问：

- (1) 该系统中，一次访存的时间下限和上限各是多少？(给出计算过程)
- (2) 若已经先后访问过 0、2 号页面，则虚地址 1565H 的物理地址是多少？(给出计算过程)

参考解答：

(1)

一次访存的时间下限是页面既在 TLB 中，也在内存中，

∴ 一次访存的时间下限为 110 (100+10) ns。

【查找快表的时间+访问实际物理地址的时间】

一次访存的时间上限是页面不在内存中，

∴ 一次访存的时间上限为 (10+100+100*10⁶+100) ns

【查找快表的时间+查找页表的时间+处理缺页中断、更新快表的时间+访问实际物理地址的时间】

(2)

页面大小为 4KB (2¹²) → 页内偏移量占 12 位，即对应 3 位十六进制数，则

虚地址 1565H 的页内偏移量为 565H，页号为 1H，页号转换成十进制为 1，

页号为 1 的页面不在内存，因此发生缺页中断。

按照 FIFO 算法，淘汰 0 号页，将 1 号页装入到 101H 块。

虚地址 → 物理地址的转换规则是偏移量不变，页号转换成物理块号。

∴ 物理地址为 101565H，前面三位十六进制数表示物理块号，后面三位十六进制数表示偏移量。

考察知识点：分页存储管理系统

在一个分页存储管理系统中，进程的逻辑地址空间占 32 页，每页 1024 字节。

系统的物理内存为 1M 字节。进程的页表以及所有的逻辑页面都已在内存中。系统有 TLB，平均命中率为 85%，一次 TLB 查找需要 20ns，一次内存访问需要 100ns。

问：

- 1) 进程的逻辑地址共几位
- 2) 进程的页表项共几项
- 3) 页表项中物理块号占几位
- 4) 进程访问一个逻辑页面的平均时间是多少

参考解答：

1)

∴ 进程的逻辑空间占 32 页，每页 1024 字节

32 × 1024 = 32768 (2¹⁵)

∴ 进程的逻辑空间占 32768 字节，
逻辑地址占 15 位

2)

进程的逻辑地址空间占 32 页，

每一页都对应一个页表项，

∴ 进程的页表项共 32 项

3)

∴进程的逻辑地址空间中每页占 $1024(2^{10})$ 字节

∴页内偏移量占10位

∴逻辑地址转换成物理地址的规则是偏移量不变，页号转换成块号

∴物理块的偏移量占10位

∴物理内存为 $1M(2^{20})$ 字节

∴物理块的位数为20位

∴物理块号的位数=物理块的位数-物理块的偏移量位数=20-10=10

4)

进程访问一个逻辑页面的平均时间=形成物理地址的时间+取操作数的时间

=快表命中的时间+未命中快表，访问页表的时间+取操作数的时间

$=\alpha \times \lambda + (1-\alpha) \times (t + \lambda) + t$

$=2t + \lambda - \alpha \times t$

【其中 t 表示访问一次内存的时间， λ 表示查快表的时间，

α 表示表示快表的命中率】

将 $t=100$ ， $\lambda=20$ ， $\alpha=0.85$ 代入上式得：

进程访问一个逻辑页面的平均时间= $200+20-0.85*100=135n$