《操作系统》作业

岳锦鹏

2024 年 3 月 29 日——2024 年 6 月 18 日

目录

第二章　进程与线程

1. 设有一台计算机，有两条 I/O 通道，分别挂一台输入机和一台打印机。若要把输入机上的数据逐一地输入到缓冲区 B1 中，然后处理，并把结果搬到缓冲区 B2 中，最后在打印机上输出。请问：

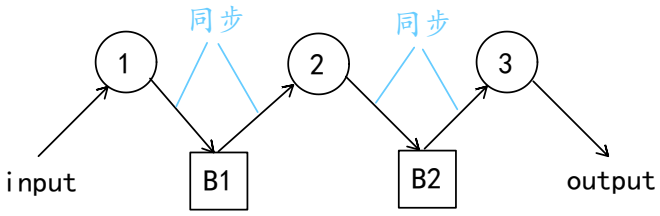
(1) 系统可设置哪些进程完成这一任务？这些进程间有什么具体制约关系？

进程一：把输入机上的数据逐一地输入到缓冲区 B1 中；
进程二：从缓冲区 B1 中取数据，处理后把结果放到缓冲区 B2 中；
进程三：从缓冲区 B2 中取数据，在打印机上输出。

这里的“逐一地”“缓冲区”等关键词表明是单个缓冲区而不是缓冲队列。那么进程一和进程二之间关于缓冲区 B1 存在同步的制约关系，进程二和进程三之间关于缓冲区 B2 存在同步的制约关系。(这里不需要互斥)

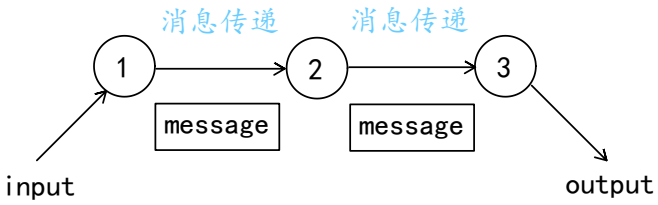
(2) 用 P-V 操作写出这些进程的同步算法。

```
struct semaphore B1_empty = 1, B1_full = 0, B2_empty = 1, B2_full = 0;
number temp1, temp2, temp3;
buffer B1, B2;
cobegin
void process1() {
    while(1) {
        temp1 = input();
        P(B1_empty);
        B1 = temp1;
        V(B1_full);
    }
}
void process2() {
    while(1) {
        P(B1_full);
        temp2 = B1_full;
        V(B1_empty);
        temp2 = processing(temp2);
        P(B2_empty);
        B2_full = temp2;
        V(B2_full);
    }
}
void process3() {
    while(1) {
        P(B2_full);
        temp3 = B2_full;
        V(B2_empty);
        output(temp3);
    }
}
coend
```



(3) 用 Send 和 Receive 原语写出这些进程的同步算法。

```
cobegin
void process1() {
    number item;
    message m;
    while (1) {
        item = input();
        m = build_message(item);
        Send(process2, &m);
    }
}
void process2() {
    number item;
    message m;
    while (1) {
        Receive(process1, &m);
        item = extract_item(&m);
        item = processing(item);
        m = build_message(item);
        Send(process3, &m);
    }
}
void process3() {
    number item;
    message m;
    while (1) {
        Receive(process2, &m);
        item = extract_item(&m);
        output(item);
    }
}
coend
```



2. “哲学家就餐问题”除课堂上介绍的方案外，有没有其他解决方案？有的话，请写出相应的解决方案。

有，比较容易想到的一种最简单的方案是，每次只允许一个哲学家就餐，也就是将所有的叉子视为一个整体的临界资源，但是缺点也很明显，会导致性能低下，因为五个叉子可以满足两个哲学家同时就餐。因此下面使用另一种方案。

大致方案是偶数序号的哲学家先拿左边的叉子，奇数序号哲学家先拿右边的叉子（释放的时候就不需要讲究先放左边还是先放后边了）。

```
struct semaphore forks[5] = {1, 1, 1, 1, 1};
void philosopher(i) {
    while (1) {
        think;
        if (i % 2) {
            P(forks[(i + 1) % 5]);
            P(forks[i])
        } else {
            P(forks[i]);
            P(forks[(i + 1) % 5]);
        }
        eat;
        V(forks[i]);
        V(forks[(i + 1) % 5]);
    }
}
```

3. a，b 两点之间是一段东西向的单行车道，现要设计一个自动管理系统，管理的规则如下：当 a，b 之间有车辆在行驶时同方向的车可以同时驶入 a，b 段，但另一个方向的车必须在 a，b

段以外等待；当 a，b 之间无车辆在行驶时，到达 a 点（或 b 点）的车辆可以进入 a，b 段，但不能同时驶入；当某方向在 a，b 段行驶的车辆驶出了 a，b 段且暂无车辆进入 a，b 段时，应让另一方向等待的车辆进入 a，b 段行驶。请用 PV 操作作为工具，对 a，b 段实现正确管理以确保行驶安全。

可能会存在一个问题，就是一个方向一直有车辆，导致另一个方向无法进入车道，从而产生很长的排队。

还有个问题，是否要考虑车辆排队的过程，即先入先出，或先排队先进入。

这里先不考虑这两个问题。

这种问题大多数做法都是使用两个互斥量和两个计数值，这里尝试使用一个互斥量和一个计数值，计数值为正表示从 a 到 b 有车辆，计数值为负表示从 b 到 a 有车辆。但是缺点是发生任何一个事件（a 或 b 方向进入或离开车辆）都需要访问临界资源（使用同一个 mutex 信号量），可能会导致性能较低。

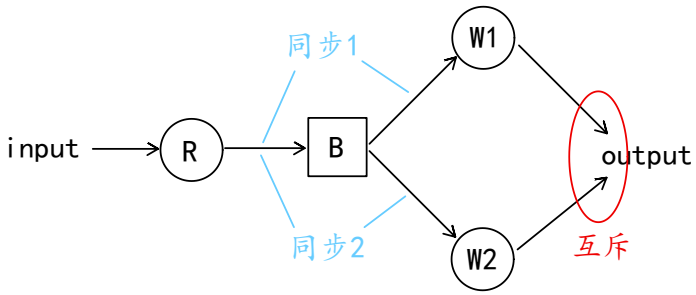
```
struct semaphore mutex = 1, s = 1;
// 车道内的车辆数
int count = 0; // 正数为从 a 到 b，负数为从 b 到 a
cobegin
void a_to_b() {
    P(mutex);
    if (count <= 0) {
        P(s);
    }
    count++;
    V(mutex);
    通过 a -> b;
    P(mutex);
    count--;
    assert(count >= 0);
    if (count == 0) {
        V(s);
    }
    V(mutex);
}
```

```
void b_to_a() {
    P(mutex);
    if (count >= 0) {
        P(s);
    }
    count--;
    V(mutex);
    通过 b -> a;
    P(mutex);
    count++;
    assert(count <= 0);
    if (count == 0) {
        V(s);
    }
    V(mutex);
}
coend
```

4. 假定有三个进程 R、W1、W2 共享一个缓冲区 B，B 中每次只能存放一个整数。进程 R 每次启动输入设备读一个整数且把它存放在缓冲区 B 中，若存放到缓冲区 B 中的是奇数，则由进程 W1 将其取出打印，否则由进程 W2 将其取出打印。要求用 PV 操作管理这 3 个并发进程，使它们能够正确同步工作。

这里使用了三个信号量实现 R 与 W1 的同步、R 与 W2 的同步，一个信号量实现了打印的互斥。

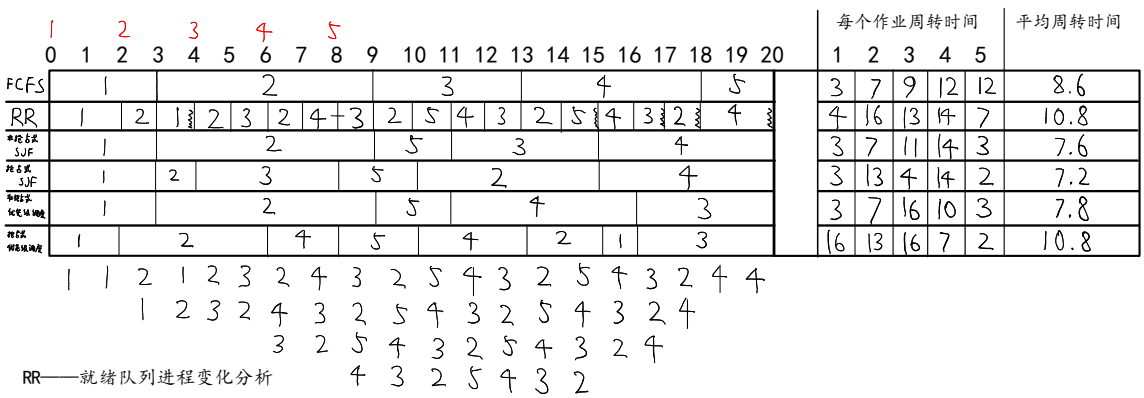
```
struct semaphore odd = 0, even = 0, empty = 1;
struct semaphore output_mutex;
buffer B;
cobegin
void R() {
    int temp;
    while (1) {
        temp = input();
        P(empty);
        B = temp;
        if (temp % 2) {
            V(odd);
        } else {
            V(even);
        }
    }
}
void W1() {
    int temp;
    while (1) {
        P(odd);
        temp = B;
        V(empty);
        P(output_mutex);
        output(temp);
        V(output_mutex);
    }
}
void W2() {
    int temp;
    while (1) {
        P(even);
        temp = B;
        V(empty);
        P(output_mutex);
        output(temp);
        V(output_mutex);
    }
}
coend
```



5. 假定在一个 CPU 上执行以下 5 个作业：（优先数小者优先级高）

作业号	1	2	3	4	5
到达时间	0	2	4	6	8
优先数	4	3	5	2	1
运行时间	3	6	4	5	2

当分别采用 FCFS、RR(时间片为 1)、非抢占式 SJF、抢占式 SJF、优先级调度 5 种调度算法时，试（1）画出调度图（2）计算每个作业的周转时间（3）计算平均周转时间。



第三章　存储管理

1. 一个 32 位的系统支持的逻辑空间最大为 2^{32} 字节，假如一个分页系统的页面大小为 4KB（即 2^{12}），一个页表项 4 个字节。请问：一个进程页表最多可有多少个表项？此时该进程的页表需要占多大的内存？

按照题意来看，页表需要放到内存中，因此 2^{32} 字节的逻辑空间不能全部用来放分页页面。那么一个页面对应一个页表项，它们需要 $2^{12}+4$ 字节，所以一个进程页表最多可有

$$\frac{2^{32}}{2^{12}+4}=\frac{1073741824}{1025}\approx 1047552.99902439 \xrightarrow{\text{向下取整}} 1047552$$

个表项。此时该进程的页表需要占 $1047552\times 4=4190208$ 个字节的内存。

2. 在分页式系统中，其页表存放在内存中

(1) 如果对内存的一次存取需要 100μs，试问实现一次页面访问至少需要存取时间是多少？

先访问一次页表，得到页面的地址以及是否缺页，此时如果没有缺页异常，直接从内存中就能访问到页面，所以实现一次页面访问至少需要存取时间为 $100+100=200\mu$s。

(2) 如果系统有快表，快表的命中率为 80%，当页表项在快表中时，其查询快表的时间可忽略不计，试问此时平均存取时间为多少？

当页表项在快表中时（并且没有缺页），此时可以认为一次页面访问只需要一次内存访问的时间，即 100μs；所以此时平均存取时间（不考虑缺页）为

$$\begin{aligned}EX&=100\mu\text{s}\times P(\text{快表命中})+200\mu\text{s}\times P(\text{快表未命中})\\&=100\mu\text{s}\times 0.8+200\mu\text{s}\times 0.2=120\mu\text{s}\end{aligned}$$

(3) 采用快表后的平均存取时间比没有采用快表时下降了百分之几？

$$1-\frac{120}{200}=\frac{2}{5}=0.4=40\%$$

下降了 40%。

3. 某虚拟存储器的用户编程空间共 32 个页面，每页 1KB，主存为 16KB。假定某时刻用户表中已调入主存的页面的虚拟页号和物理块号对照表为表一，则与逻辑地址相对应的物理地址为表二。

虚拟页号	物理块号
0	5
1	10
2	4
8	7

表一

逻辑地址	物理地址
0A5CH	A( )
1A5CH	B( )
925DH	C( )

表二

可供选择的答案：A，B，C：（1）缺页（2）1E5DH（3）2A5CH（4）115CH（5）125CH（6）165CH（7）越界（8）以上答案都不对

页面大小 1KB，也就是 2^{10} 字节（其实准确说 1KiB 才是 2^{10} 字节，1KB 是 10^3 字节），也就是逻辑地址的后 10 位代表页内偏移。

由于一共 $32 = 2^5$ 个页面，所以（序号从右数 0 开始）第 10 到 14 位是页号，题目中的逻辑地址是 4 位十六进制数（最后的 H 表示这是十六进制数），也就是 16 位二进制数，因此第 15 位（最高位）必定为 0，否则就是虚拟页号越界。以 0A5CH 为例：

$$0|\underbrace{000\ 10}_{\text{虚拟页号}}|\underbrace{10\ 0101\ 1100}_{\text{页内偏移}}$$

从逻辑地址转换到物理地址，只需要把虚拟页号换成对应的物理块号即可，这里虚拟页号为二进制的 00010 也就是十进制的 2，对应的物理块号为 4，也就是二进制的 00100，转换后为

$$0|\underbrace{001\ 00}_{\text{物理块号}}|\underbrace{10\ 0101\ 1100}_{\text{页内偏移}}$$

转换成十六进制也就是 125CH，所以 A(5)。

同理，对于 1A5CH，

$$0|\underbrace{001\ 10}_{\text{虚拟页号}}|\underbrace{10\ 0101\ 1100}_{\text{页内偏移}}$$

虚拟页号 00110，即十进制的 6，在表一中没有，所以是缺页，从而 B(1)。

对于 925DH

$$1|\underbrace{001\ 00}_{\text{虚拟页号}}|\underbrace{10\ 0101\ 1101}_{\text{页内偏移}}$$

最高位为 1，根据前面的分析，虚拟页号越界，所以 C(7)。

综上所述，A(5), B(1), C(7)。

4. 分页存储管理系统中，逻辑地址长度为 16，页面大小为 4K。一个进程有 6 个页面，且页号为 1、2、3 的页面依次存放在物理块 5、10、11 中，试问：

4K 看作是 2^{12}，也就是 12 位逻辑地址，也就是右起 3 位十六进制。逻辑地址长度为 16，$16 - 12 = 4$ 位表示页号，所以左起 1 位十六进制表示页号。有 6 个页面，所以页号从 0 到 5 有效，否则越界。

(1) 逻辑地址 4010、8500 和 25000 所对应的物理地址分别为多少?

这里的数字没有字母后缀，应该是十进制，需要先转化成十六进制。

十进制逻辑地址	十六进制逻辑地址	物理地址
4010	0|FAA	缺页
8500	2|134	A134
25000	6|1A8	越界

(2) 逻辑地址 2F6AH、0578H 和 6ACDH 所对应的物理地址分别为多少?

逻辑地址	物理地址
2|F6A H	AF6A H
0|578 H	缺页
6|ACD H	越界

5. 假定某采用页式存储管理的系统中，主存容量为 1M，被分成 256 块，块号为 0、1、2、…、255。现有一个共 4 页（页号为 0、1、2、3）的作业被依次装入到主存的块 2、4、1、5 中。请回答:

(1) 主存地址应该用多少位来表示?

1M 省略了单位，应该是指 1MB，看作是 2^{20} 字节，由于寻址的单位为字节，所以主存地址应该用 20 位来表示。

(2) 作业每一页的长度为多少字节? 逻辑地址中的页内地址部分应占用多少位?

$1\text{MB} = 2^{20}$ 字节，分成 $256 = 2^8$ 块，所以每块的大小为 $\frac{2^{20}}{2^8} = 2^{12}$ 字节，块大小和页大小一样，因此作业每一页的长度为 2^{12} 字节，逻辑地址中的页内地址部分应占用 12 位。

(3) 把作业中每一页占用的主存块起始地址填入下表。

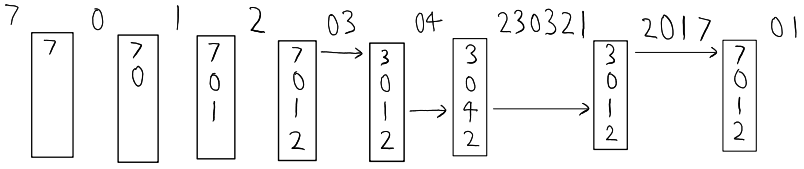
页号	起始地址
0	02 000 H
1	04 000 H
2	01 000 H
3	05 000 H

(4) 若作业执行中要从第 0 页的第 75 单元和第 3 页的第 548 单元读信息，那么，实际应从主存的哪两个单元读信息？请把应访问的主存绝对地址用二进制编码的十六进制数表示。

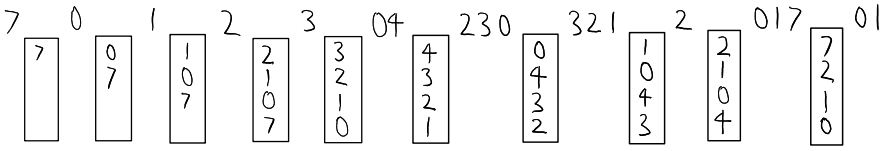
这里“二进制编码的十六进制数”表述不清，直接描述成“十六进制”或许更好。第 0 页的第 75 单元为 0x02 048，第 3 页的第 548 单元为 0x05 224。

6. 在请求式分页系统中，运行一个共有 8 页的作业，且作业在主存中分配到 4 块主存块，作业执行时访问页面顺序为 7，0，1，2，0，3，0，4，2，3，0，3，2，1，2，0，1，7，0，1，假如最初四个页面通过缺页中断装入，请问用最佳置换算法、先入先出（FIFO）算法和最近最久未使用（LRU）算法时，它们的页面缺页和置换次数分别为多少？（要求画出页面置换图）

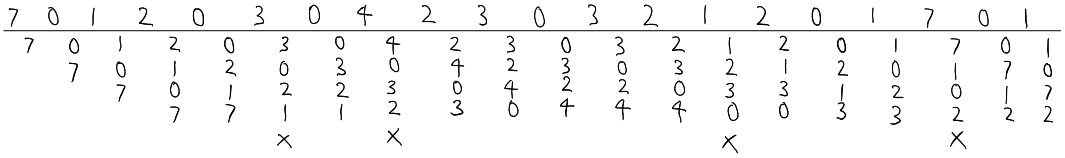
Opt



FIFO



LRU



（图中标了 × 的是发生置换的）

页面置换算法	页面缺页次数	置换次数
最佳置换（Opt）	8	4
先入先出（FIFO）	10	6
最近最久未使用（LRU）	8	4

第四章　文件系统

1. 一个文件系统的盘块大小为 1KB，每块地址用 4B 表示，当分别采用连续、链接、二级索引和 UNIX S V 分配方案时，试确定各方案能管理的最大文件，管理一个 10MB 的大文件和一个 10KB 的小文件时所需的管理专用块数，以及要访问大文件的第 9M+3.5KB 单元时需要的磁盘 I/O 操作次数。

   A∼D: (1) 不受限制 (2) 1KB (3) 256KB (4) 64MB (5) 16GB (6) 1KB+256KB+64MB+16GB (7) 10KB+256KB+64MB+16GB

二级索引管理的最大文件:

$$256 \times 256 \times 1\text{KB} = 2^8 \times 2^8 \times 1\text{KB} = 2^{16}\text{KB} = 2^6\text{MB} = 64\text{MB}$$

二级索引专用块数:

$$10\text{KB}/1\text{KB} = 10\text{个存储块}$$

$$\lceil 10/256 \rceil = 1\text{个二级索引块}, \quad \lceil 1/256 \rceil = 1\text{个一级索引块}$$

$$10\text{MB}/1\text{KB} = 10 \times 2^{10}\text{个存储块}$$

$$10 \times 2^{10}/256 = 10 \times 2^2 = 40\text{个二级索引块}, \quad \lceil 1/256 \rceil = 1\text{个一级索引块}$$

UNIX 专用块数:

$$10\text{MB} - 10\text{KB} - 256\text{KB} = (10 \times 2^{20} - 10 \times 2^{10} - 256 \times 2^{10}) = 10213376\text{B} = 9974\text{KB}$$

$$\lceil 9974/256 \rceil = 39\text{个二级索引块}, \quad \lceil 39/256 \rceil = 1\text{个一级索引块}$$

$$1 + 39 + 1 = 41\text{个管理专用块}$$

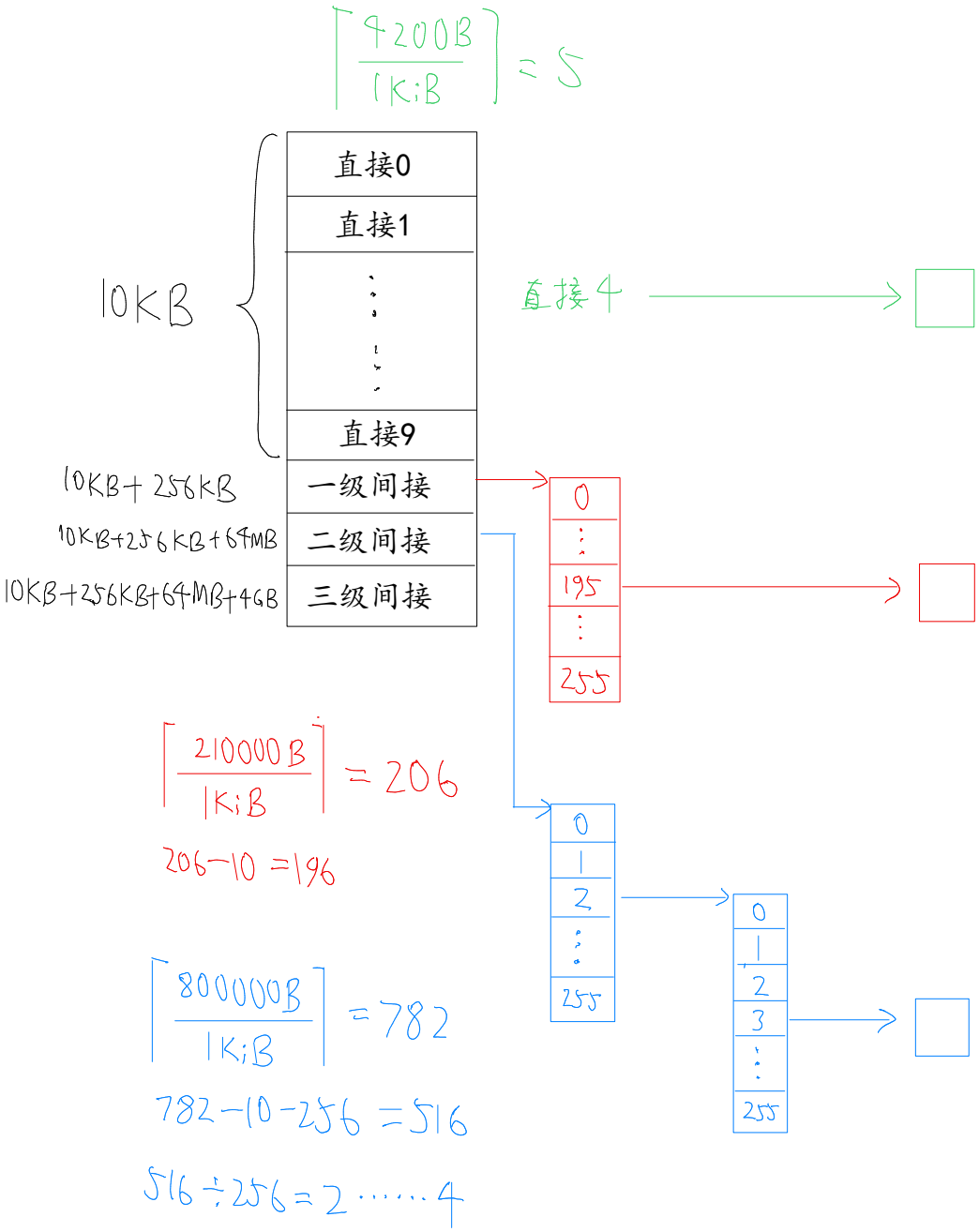
大文件的某处信息需要的 IO 次数:

$$\lceil 9 \times 2^{10} + 3.5 \rceil + 1 = 9221$$

		连续分配	链接分配	二级索引	UNIX
管理的最大文件		A(1)	B(1)	C(4)	D(7)
管理用的专用块数	10KB 文件	0	0	2	0
	10MB 文件	0	0	41	41
大文件的某处信息	9M+3.5KB	1	9221	3	3

2. 在 Unix 文件系统中，文件的物理组织为 Unix 直接间接混合寻址方式，假设一个进程要在第 4200 字节、第 210000 字节和第 800000 字节三个偏移处读文件，请问分别要访问多少次磁盘？并以必要的图示说明访问之过程。假设该文件的 FCB（即文件说明或文件控制块）已读入内存，每个磁盘块大小为 1K，块号用 32 位的指针表示。

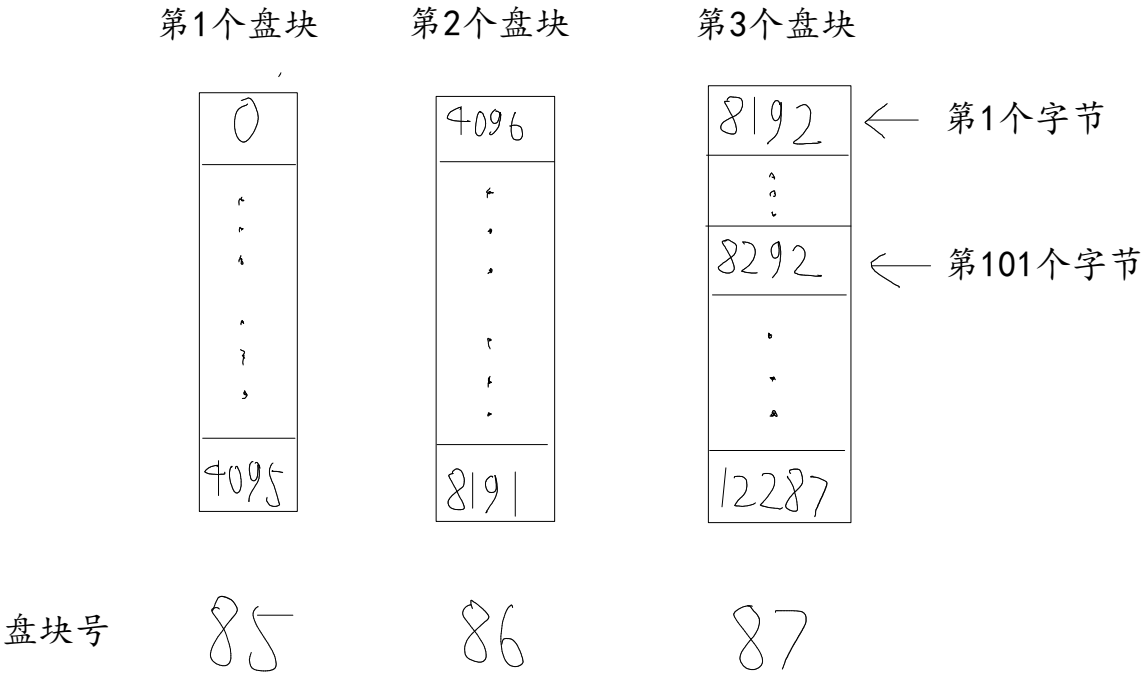
32 位即 4B，所以一个索引块能存储 1KB/4B = 256个块指针。



图中用不同颜色写出了不同偏移处的计算步骤以及访问过程。访问磁盘次数即不同颜色的箭头数量，在第 4200 字节、第 210000 字节和第 800000 字节三个偏移处读文件，分别要访问 1、2、3 次磁盘。

3. 假设文件系统的盘块大小为 4KB, 某文件的物理结构采用连续文件方式，假设该文件的首个盘块的盘块号为 85，那么该文件的第 8292 字节单元在第几个盘块上? 其盘块号为多少? 该字节单元是盘块内的第几字节?

$$8292\text{B}/4\text{KB} = 8292 \div (4 \times 2^{10}) = 2 \cdots\cdots 100$$



所以该文件的第 8292 字节单元在第 3 个盘块上，其盘块号为 87，该字节单元是盘块内的第 101 字节。

第五章　I/O 设备管理

1. 假定某磁盘共有 200 个柱面（编号为 0~199），如果在为访问 80 号柱面的请求者服务后，当前正在为访问 108 号柱面的请求者服务，同时有若干个请求者在等待服务，它们依次要访问的柱面号为：

   187，64，169，48，171，118，120，84

   (1) 分别用先来先服务（FCFS）、最短寻道时间优先（SSTF）、扫描（SCAN）和循环扫描（CSCAN）算法进行磁盘调度时，试确定实际的服务次序。

   (2) 按实际服务次序计算（1）中四种算法下磁臂移动的距离。

80→108	服务次序	磁臂移动的距离
FCFS	187 (123) 64 (105) 169 (121) 48 (137) 171 (53) 118 (2) 120 (36) 84	577
SSTF	118 (2) 120 (72) 84 (64) 48 (139) 169 (139) 171 (187)	213
SCAN	118 (69) 120 (69) 169 (171) 187 (139) 84 (139) 64 (48)	208
CSCAN	118 (69) 120 (69) 169 (171) 187 (0) → 48 (36) 64 (84)	105

每次磁臂移动的距离已经用蓝色字标在两次访问的柱面号之间，要注意循环扫描的返回是瞬间完成，不应算到磁臂移动的距离中。

第六章　死锁

1. 在某系统中，有 N 个进程共享 R 台同类设备资源，每个进程最多需要 M 台设备资源，试问：N 最多为几时才能保证系统不会发生死锁？请简略说明原因。

（这里的 N、R、M 应该均为正整数）

当 N 固定时，根据抽屉原理，每个进程都恰好分到最大资源数量少一个资源，并且都申请一个资源，这时如果没有资源可以分配，那么就死锁了；如果有至少一个资源可以分配，那么就不会死锁。所以 R 最少应取 $N \times (M-1)+1$ 才能保证不会发生死锁，那么

$$R \geqslant N \times (M-1)+1 \implies N \leqslant \frac{R-1}{M-1}$$

因为 N 为整数，所以 N 最多为 $\left\lfloor \frac{R-1}{M-1} \right\rfloor$。

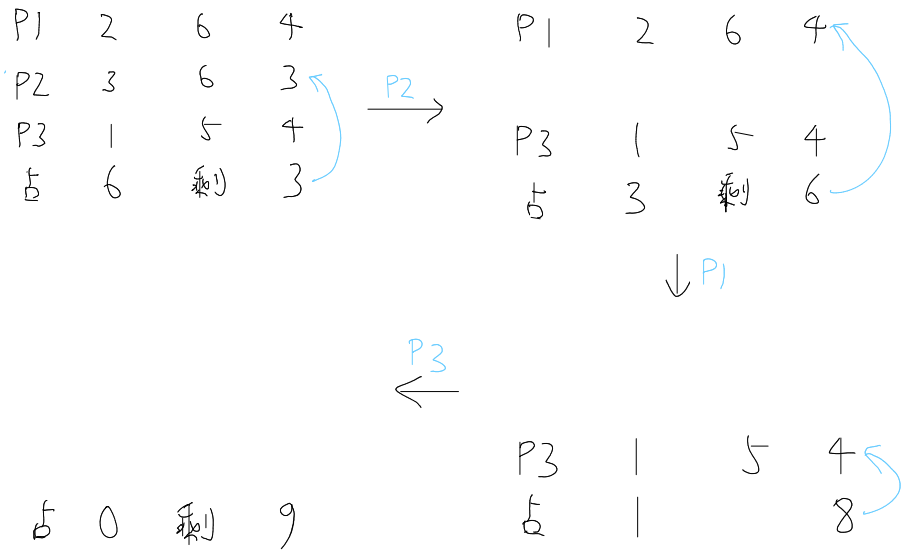
2. 考虑有 3 个进程共享 9 个资源，当前资源分配情况如下：

进程	已占资源数	最大需求量
P1	2	6
P2	3	6
P3	1	5

请回答以下问题：

(1) 目前系统是否处于安全状态？为什么？

目前已占资源总数为 6，那么还有 3 个资源，这时如果分配给 P2，P2 就达到了最大需求量，那么 P2 运行结束后就有 6 个空闲资源，这时可以分配给 P1 4 个资源，P1 就达到了最大需求量，P1 运行结束后分配给 P3 4 个资源，P3 就达到了最大需求量，P3 运行完后就结束了。**所以可以找到资源分配安全序列 P2、P1、P3，所以目前系统处于安全状态。**示意图如下：



(2) 如果接着 3 个进程均再申请 2 个资源，可以先分配资源给哪个进程?

在未实际分配资源时，安全序列不变，仍然为 P2、P1、P3，所以可以先分配资源给 **P2**。

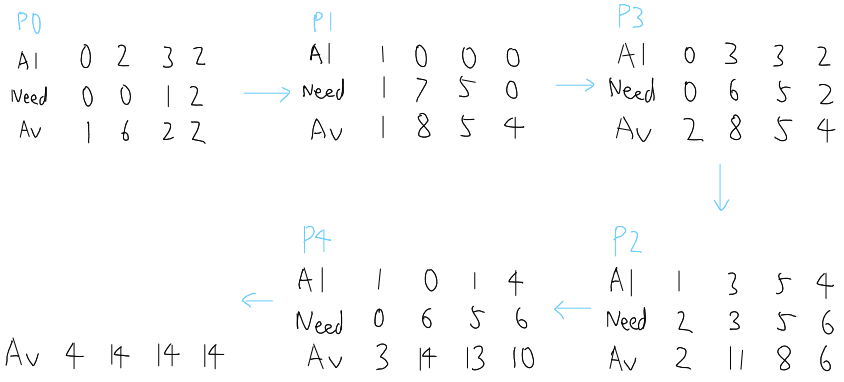
3. 假如系统中有 5 个进程 {P0，P1，P2，P3，P4} 和 4 种类型资源 {A，B，C，D}，T0 时刻系统的资源分配情况如下所示：

进程	Al				Need				Av			
P0	0	2	3	2	0	0	1	2	1	6	2	2
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	3	3	2	0	6	5	2				
P4	1	0	1	4	0	6	5	6				

试问：

(1) T0 时刻该系统是否安全?

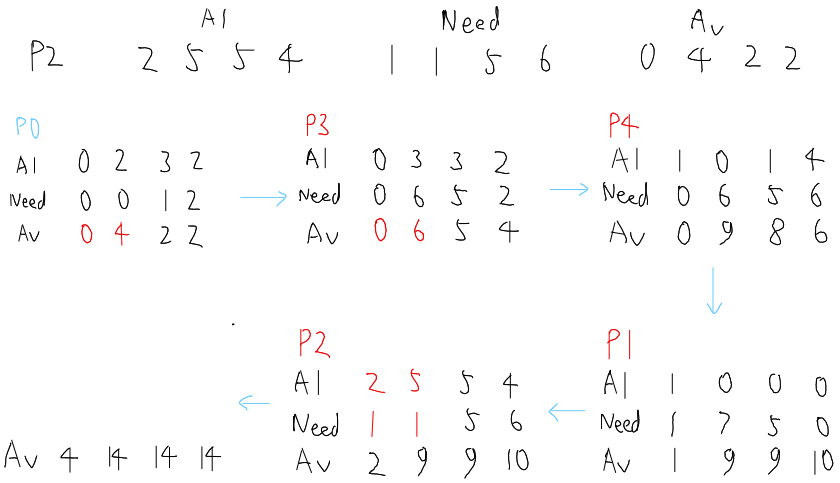
尝试寻找安全序列如下图所示：



成功找到安全序列 P0、P1、P3、P2、P4，所以 T0 时刻系统**安全**。

(2) T1 时刻进程 P2 提出资源请求 Re2(1，2，0，0)，能否将资源分配给它?

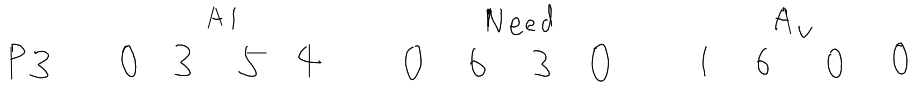
Re2 < Av，进行试探性分配，尝试寻找安全序列如下图所示：



与 T0 时刻相比有不同的地方已用红色标出，所以成功找到安全序列 P0、P3、P4、P1、P2，所以**能**将资源分配给它。

(3) T2 时刻进程 P3 提出资源请求 Re3(0，0，2，2)，能否将资源分配给它?

Re3 < Av，进行试探性分配，尝试寻找安全序列如下图所示：



发现找不到安全序列，所以**不能**将资源分配给它。