



# 例1

某单道程序设计系统中，5个作业A、B、C、D、E先后到达系统，其到达时间及运行时间如下表所示：分别计算用FCFS、SJF和高响应比优先算法调度时的平均作业周转时间和平均带权作业周转时间（忽略进程切换的开销，且假定所有作业都是CPU密集型作业。结果保留一位小数）。

作业	到达时间	运行时间
A	0	3
B	2	5
C	3	3
D	4	4
E	5	1

- 
- 采用FCFS算法时:

$$T=(3+6+8+11+11)/5=7.8$$

$$W=(3/3+6/5+8/3+11/4+11/1)/5=3.7$$

- 采用SJF算法时:

$$T=(3+14+3+7+2)/5=5.8$$

$$W=(3/3+14/5+3/3+7/4+2/1)/5=1.7$$

- 采用高响应算法:

$$T=(3+6+9+12+4)/5=6.4$$

$$W=(3/3+6/5+9/3+12/4+4/1)/5=2.4$$



## 例2

---

一台计算机有10台磁带机被 $n$ 个进程竞争，每个进程最多需要3台磁带机，那么 $n$ 最多为\_\_\_\_\_时，系统没有死锁的危险？

解： $n$ 最大为4。




## 例3

在银行家算法中，若出现下述的资源分配情况：

Process	Max	Allocation	Available
P0	0 0 4 4	0 0 3 2	1 6 2 2
P1	2 7 5 0	1 0 0 0	
P2	3 6 10 10	1 3 5 4	
P3	0 9 8 4	0 3 3 2	
P4	0 6 6 10	0 0 1 4	

试问：

- 1) 该状态是否安全？
- 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？
- 3) 如果系统立即满足P2的上述请求，系统是否立即进入死锁状态？




解：

- 1) 利用安全性算法对上面的状态进行分析  
(如下表所示)，找到了一个安全序列 {P0, P3, P4, P1, P2} 或 {P0, P3, P1, P4, P2}，故系统是安全的。



资源情况 进程	Work	Need	Allocatio n	Work+Alloc ation	Finis h
	A B C D	A B C D	A B C D	A B C D	
P0	1 6 2 2	0 0 1 2	0 0 3 2	1 6 5 4	True
P3	1 6 5 4	0 6 5 2	0 3 3 2	1 9 8 6	True
P4	1 9 8 6	0 6 5 6	0 0 1 4	1 9 9 10	True
P1	1 9 9 10	1 7 5 0	1 0 0 0	2 9 9 10	True
P2	2 9 9 10	2 3 5 6	1 3 5 4	3 12 14 14	True



2) P2发出请求向量Request (1, 2, 2, 2) 后, 系统按照银行家算法进行检查:

$\text{Request}_2(1, 2, 2, 2) \leq \text{Need}_2(2, 3, 5, 6)$  ;

$\text{Request}_2(1, 2, 2, 2) \leq \text{Available}(1, 6, 2, 2)$  ;

系统先假定可为P2分配资源, 并修改Available, Allocation<sub>2</sub>和Need<sub>2</sub>向量:

Availabe= (0, 4, 0, 0) Allocation<sub>2</sub>= (2, 5, 7, 6)

Need<sub>2</sub>=(1, 1, 3, 4)

进行安全性检查: 此时对所有进程, 条件 $\text{Need}_i \leq \text{Available}(0, 4, 0, 0)$  都不成立, 即Available不能满足任何进程的请求, 故系统进入不安全状态。因此, 当进程P2提出请求Request (1, 2, 2, 2) 后, 系统不能将资源分配给它。

3) 系统立即满足进程P2的请求 (1, 2, 2, 2) 后, 并没有马上进入死锁状态。因为, 此时上述进程并没有申请新的资源, 并未因得不到资源而进入阻塞状态。只有当上述进程提出新的请求, 并导致所有没执行完的多个进程因得不到资源而阻塞时, 系统才进入死锁状态。



## 例4


•1、某32位请求分页存储管理系统中，页内地址占12位，进程P的页面信息如下表所示：

（1）该系统的页面大小是多少？地址空间最多允许多少页？

（2）给定逻辑地址为9000B，则其页号，页内地址（位移量）和其对应的物理地址是多少？

页号	块号
0	5
1	2
2	1
3	4





答案：（1）、页面大小4KB，最多可以有1M个页

（2）、逻辑地址9000在2号页面，查表知道内存的物理块是1，页内偏移808B  
物理地址是 $4096+808=4904B$



## 例5

---

对进程访问页面顺序为1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5, 指出在驻留集大小分别为3、4时, 使用FIFO替换算法的缺页次数和缺页率。结果说明了什么?

# 先进先出 (FIFO) 页面置换算法 (续)

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

4 frames

1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

FIFO Replacement – **Belady's Anomaly**

– more frames  $\Rightarrow$  less page faults



## 例6

在一个请求分页系统中，有一用户作业，它依次要访问的字地址序列是：114，218，120，46，158，440，102，323，432，260，360，167。现分配给该作业的主存共300字，页的大小为100字，试问采用最佳（OPT）、先进先出（FIFO）、最近最久未使用（LRU）三种置换算法的缺页次数各是多少？（注意：作业的主存空间最初都是空的，凡第一次用到的页面都产生一次缺页

答：由于页的大小为100字，则分配给作业300字内存对应的页面数 $M=3$ ，且该作业的页面走向为：

1，2，1，0，1，4，1，3，4，2，3，1

故可计算得：

最佳（OPT）调度算法将产生6次缺页中断。

先进先出（FIFO）调度算法将产生7次缺页中断。

最近最久未使用（LRU）调度算法将产生7次缺页中断。



# 例7

假定在某移动臂磁盘上，刚刚处理了访问145号磁道的请求，目前正在为访问126号磁道的请求服务，同时有若干请求者在等待服务，它们依次访问的磁道号为

86, 147, 91, 177, 94, 150, 120, 175, 130

- (1) 请写出用先来先服务算法作为磁道调度算法时，实际服务次序，并计算平均寻道长度（四舍五入保留1位小数）。
- (2) 请写出用最短寻道时间优先算法作为磁道调度算法时，实际服务次序，并计算平均寻道长度（四舍五入保留1位小数）。
- (3) 请写出用扫描（电梯）算法作为磁道调度算法时，实际服务次序，并计算平均寻道长度（四舍五入保留1位小数）。



先来先服务算法:

服务次序: 86, 147, 91, 177, 94, 150, 120, 175, 130

移动距离: 40, 61, 56, 86, 83, 56, 30, 55, 45

平均寻道长度为: 56.9

最短寻道时间优先算法

服务次序: 130, 120, 147, 150, 175, 177, 94, 91, 86

移动距离: 4, 10, 27, 3, 25, 2, 83, 3, 5

平均寻道长度为: 18.0

扫描(电梯)算法

服务次序: 120, 94, 91, 86, 130, 147, 150, 175, 177

移动距离: 6, 26, 3, 5, 44, 17, 3, 25, 2

平均寻道长度为: 14.6



## 例8

某段表内容如下：

逻辑地址（2，14）、（3，25）、（4，10）所对应的物理地址是多少？

段号	段首地址	段长度
0	120	40
1	760	30
2	480	20
3	370	20

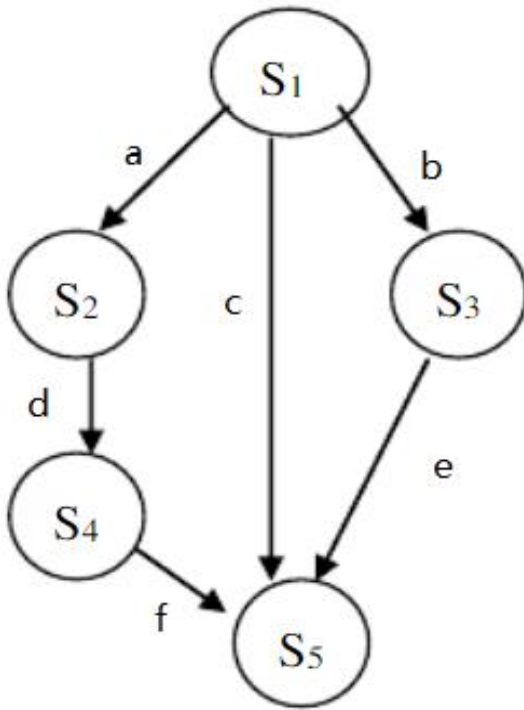
逻辑地址（2，14）的物理地址为 $480+14=494$

逻辑地址（3，25）访问越界

逻辑地址（4，10）访问越界

## 例9


请用信号量实现右图的前趋关系，箭头线上为信号量。



```
p1(){S1:signal(a);signal(b);signal(c)}  
p2(){wait(a);S2:signal(d);}  
p3(){wait(b);S3:signal(e);}  
p4(){wait(d);S4:signal(f);}  
p5(){wait(c); wait(e); wait(f);S5}
```

```
main(){  
    Semaphore a,b,c,d,e;  
    a.value=b.value=c.value=d.value  
    =e.value=0  
    cobegin  
        p1();p2();p3();p4();p5();  
    coend  
}
```





```
p1(){S1:signal(a);signal(b);signal(c)} (1分)
p2(){wait(a);S2:signal(d);} (1分)
p3(){wait(b);S3:signal(e);} (1分)
p4(){wait(d);S4:signal(f);} (1分)
p5(){wait(c); wait(e); wait(f);S5} (1 分 )
main(){
    Semaphore a,b,c,d,e;(0.5)
    a.value=b.value=c.value=d.value=e.value=0
    (0.5)
    cobegin
        p1();p2();p3();p4();p5();(1分)
    coend
}
```