**Operating System Review Problems**

1. **What is a process？ Please draw a process state transition diagram with nine states in Unix SVR4?**

进程是一个正在执行的程序（a program in execution）；九状态转换图如下所示：

fork

返回用户模式 存储器空间足够 存储器空间不足

抢占 换出

返回

重新调度进程 换入

系统调用，中断

唤醒 唤醒

退出 睡眠

中断，中断返回

换出

1. **Illustrated some scenarios（场景） where process switch（进程切换） occur?**
2. 时钟中断②i/0中断③内存失效④陷阱⑤系统调用
3. **Suppose there are three processes running in an operating system with multithreading as following:**

|  |  |
| --- | --- |
| **Process ID** | **Threads** |
| **A** | **T11, T12** |
| **B** | **T21, T22** |
| **C** | **T31, T32** |

**If the multithreading is supported by user-level（用户级） implementation(实现) and thread T11 is blocked, which thread may be scheduled to run next?（one of T21,T22,T31,T32）**

**If the multithreading is supported by kernel-level （核心级）implementation, which thread may be scheduled to run next? （one of T12,T21,T22,T31,T32）**

原因：核心级的实现，os才知道A中有T12

**4. What is race condition?**

答：竞争条件是在当多个进程或线程访问共享资源时，其最终结果依赖于多个进程的指令执行顺序。

**Give a program implementation to demonstrate（演示，展示） the mutual exclusion enforced by Test and Set (or exchange) instruction or by semaphore?**

用信号量实现互斥：

semaphore mutex;

mutex=1;

…

semwait(mutex);

…

Semsignal(mutex);

…

TS实现互斥：

Lock DB 0

…

entercritical:

TS reg,lock

CMP reg,o

JNZ entercritical

…

exitcritical:

MOV lock,0

exchange实现互斥：

Lock DB 0

…

entercritical:

MOV reg,!0

Xchg lock,reg

CMP reg,0

JNZ entercritical

…

exitcritical:

MOV lock,0

**5.Consider the following program:**

**boolean blocked[2];**

**int turn;**

**void P(int id)**

**{**

**while (true) {**

**blocked[id] = true;**

**while (turn != id){**

**while (blocked[1-id]) /\* do nothing \*/;**

**turn = id;**

**}//循环结束，则某一进程进入临界区**

**/\* critical section \*/临界区代码**

**blocked[id] = false;**

**/\* remainder \*/**

**}**

**}**

**void main()**

**{假设2个进程访问资源，我们对其行为纠错**

**blocked[0] = false;**

**blocked[1] = false;**

**turn = 0;**

**parbegin(P(0), P(1));创建2个进程**

**}**

This software segment is the solution to the mutual exclusion problem for two processes. Find a counterexample(反例) that demonstrates that this solution is incorrect.

考虑进程0执行到blocked[id] = true这个语句是发生了进程切换，操作系统调度进程1运行。这种情况下会导致进程0和进程1同时进入临界区。

**6.Give a solution（写代码） to bounded（缓冲池缓冲区有限） producer-consumer problem with semaphore.**

/\*program boundedbuffer\*/

const int sizeofbuffer=/\*buffer size\*/;

semaphore s=1;

semaphore n=0;

semaphore e=sizeofbuffer;

void producer()

{

while(true)

{

produce();

semWait(e);

semWait(s);

append();

semSignal(s);

semSignal(n);

}

}

void consumer()

{

while(true)

{

semWait(n);

semWait(s);

take();

semSignal(s);

semSignal(e);

consume();

}

}

void main()

{

Parbegin(producer,consumer);

}

**7.What is deadlock?** (死锁，4种策略，4个条件)

一组竞争系统资源或互相通信的进程间相互的“永久”阻塞，没有有效的通用解决方案，涉及到2个或更多的进程之间因对资源的需求所引起的冲突。

4种策略：忽略，预防，避免，检测与破坏

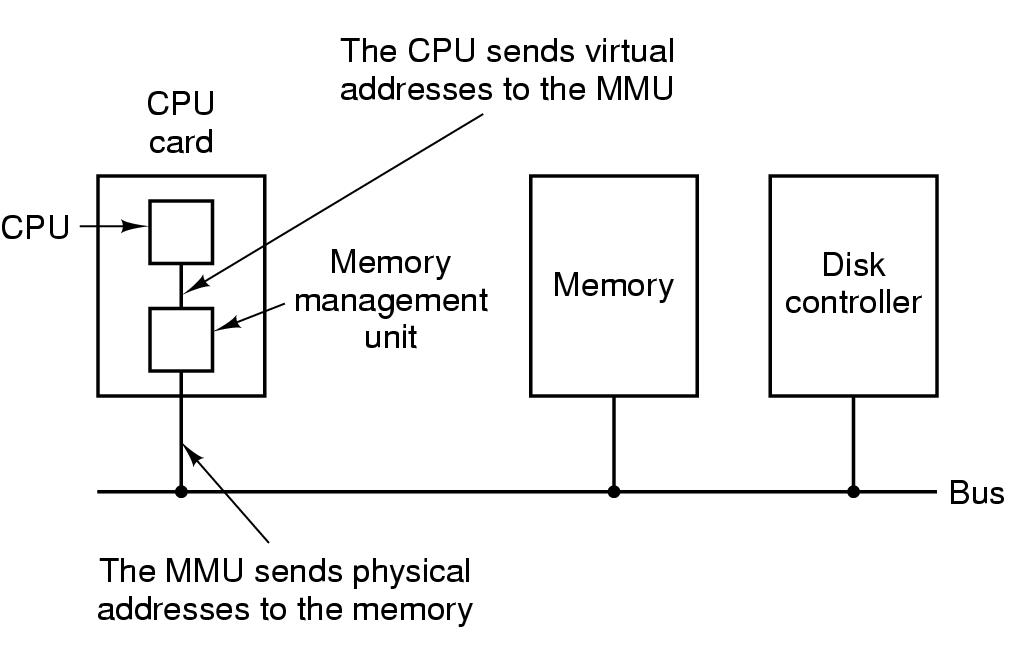
4个条件：互斥，占有且等待，非抢占，循环等待

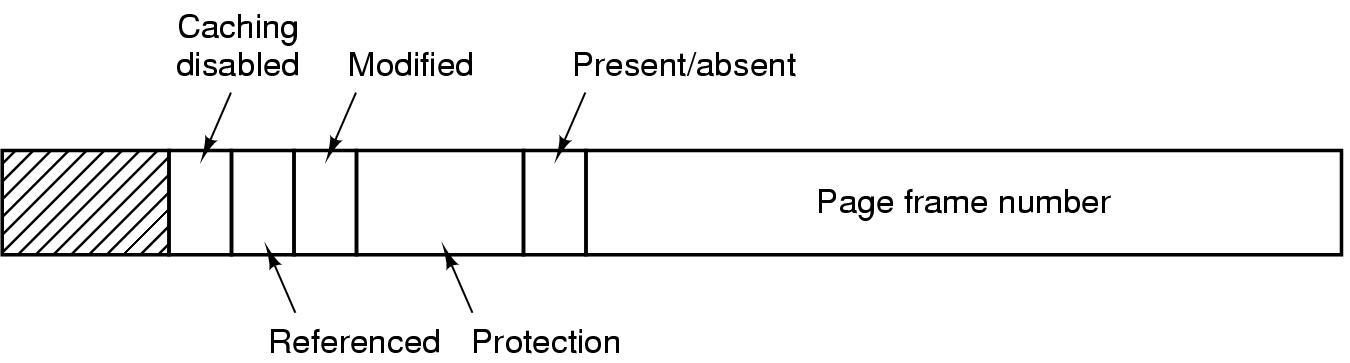
**8.Banker algorithms? （银行家策略）Safe state/unsafe state**

银行家策略即资源分配拒绝策略。一个系统有固定数目的进程和资源，任何时候一个进程可能分配到零个或多个资源。该策略确保系统中进程和资源总是处于安全状态。当进程请求一组资源时，假设同意该请求，从而改变了系统的状态，然后确定其结果是否还处于安全状态。如果是，同意这个请求；如果不是，阻塞该进程直到同意该请求时系统仍是安全的。

安全状态是指至少有一个进程执行序列不会导致死锁。不安全状态就是指一个不安全的状态，但不一定导致死锁。

**9. mapping（地址映射） in paging（分页） system:**





例题：一页4K，页号为1的页，其页框号为6，求mov AX，[10000]后页实地址？

公式Page no=(virtual address)/(page size) =10000/4096=2

公式offset=(virtual addumberress)mod(page size)=1808

公式Physical address= (Page frame number)\* (page size)+offset=6\*4k+1808=26384

Page frame number可由Page no来查找，提上会给相关信息。

**Address Translation(Mapping)地址映射**

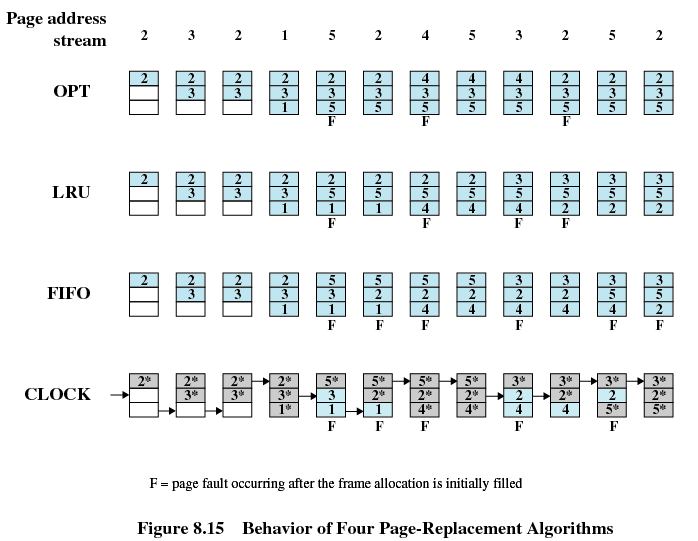
**10.What is TLB?**

**转移后备缓冲器 Translation Lookaside Buffer**

每个虚存可能引起两次物理内存的访问。一次取相应的页表项，一次取需要的数据。会导致访问时间加倍。

为克服此问题，一个特殊的高速缓存为页表项使用，它被称为转移后备缓冲器（TLB）。

**11.Page replacement algorithms （页面替换策略）(Optimal/LRU/FIFO/CLOCK) in paging system.**



LRU:

* Replaces the page that has not been referenced for the longest time

替换主存中上次使用距当前最远的页

* By the principle of locality, this should be the page least likely to be referenced in the near future根据局部性原理，这也是最近最不可能访问到的页
* Each page could be tagged with the time of last reference. This would require a great deal of overhead 每一页添加一个最后一次访问的时间标签，就需要额外的开销

CLOCK:

* Additional bit called a use bit附加位称为使用位
* When a page is first loaded in memory, the use bit is set to 1

当某一页首次装入主存中时，该帧的使用位设置为1

* When the page is referenced, the use bit is set to 1

当该页随后被访问到时，它的使用位也被置为1

* When it is time to replace a page, the first frame encountered with the use bit set to 0 is replaced.

当需要替换一页时，选择遇到的第一个使用位为0的帧替换。

* During the search for replacement, each use bit set to 1 is changed to 0

在查找替换页的时候，所有使用位为1的帧被重置为0.

**12.What is preemptive or nonpreemptive scheduling decision mode?**

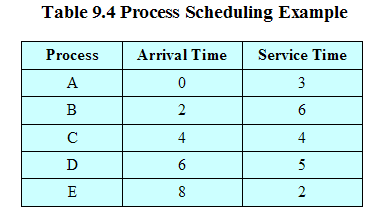
1、preemptive抢占：一旦进程处于运行状态，它就不断执行直到终止。或者因为等待I/O，或者因为请求某些操作系统服务而阻塞自己。

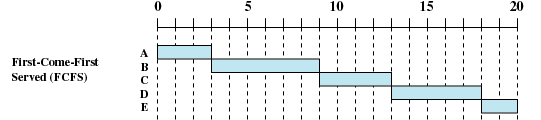
2、nonpreemptive非抢占：当前正在运行的进程可能被操作系统中断，并转移到就绪态。

3、与非抢占策略相比，抢占策略可能导致较大的开销，但是可能对所有进程提供较好的服务。

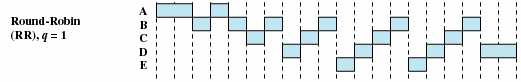
**How to compute turnaround time in different short-term scheduling algorithms?、**

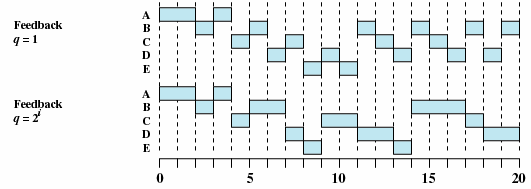
turnaround time周转时间(Tr)=等待时间+服务时间Ts（也即完成时间-到达时间）





ABCDE周转时间分别为：3,7,9,12,12其它调度策略的图英文版P403，中文版P289





基于优先级：

**13.Why Disk cache is usually used in operating system?**

一些写出的数据也许下次会被访问到，使这些数据能迅速地从软件设置的磁盘高速缓存中取出，而不是缓慢的从磁盘中取出。

**14.Disk arm scheduling algorithms磁盘调度算法（11章）**

1、先进先出（FIFO）

2、SCAN扫描

3、C-SCAN循环扫描

4、RAID独立磁盘冗余阵列

**15.What is Absolute path or relative path?**

Absolute path绝对路径：从根目录开始做起点的路径。

relative path相对路径：从当前目录开始做起点的路径。

**Explain how file pathname /usr/ast/mbox is be parsed（解析） in Unix SVR4?**

路径解析：

6 usr

1、从根目录读内容，取出目录项

2、读索引结点i—node6的内容，从属性内容中判断出usr为目录

3、从索引结点i—node6中找到第一个块地址132

26 ast

4、读地址为132的块信息，取出目录项

5、读索引结点i—node26的内容，从属性内容中判断出ast为目录

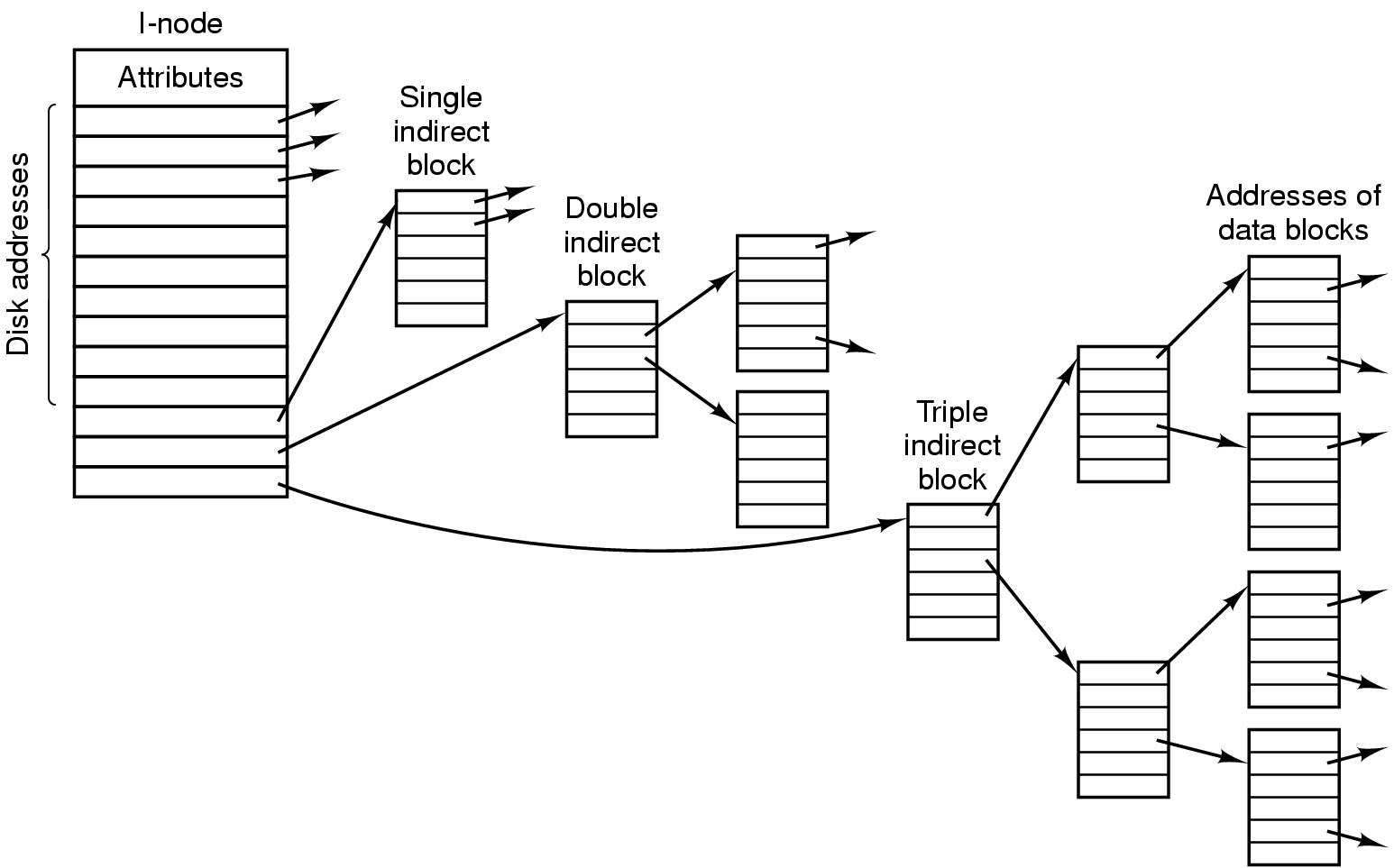
6、从索引结点i—node26中找到第一个块地址406

60 mbox

7、读地址为406的块信息，取出目录项

8、读索引结点i—node406的内容，从属性内容中判断出mbox为文件，结束搜索

**16.Consider a UNIX-style i-node with 10 direct pointers, one single-indirect pointer, and one double-indirect pointer. Assume that the disk block size is 1 Kbytes, and that the size of a disk block address is 4 bytes. How large a file can be indexed using such an i-node?**



disk block size is 1 Kbytes:一块1KB

4 bytes=4\*8=32位，所以一级间接是2^32块，二级则平方一次即可。

**文件的容量如下表：**

级 块数 字节数

直接 10 10KB

一级间接 2^32=256 256KB

二级间接 256\*256=64k 64MB