选择题

(2014).（1分x10=10分）

1．下列关于操作系统的四种陈述中，正确的是：___B___。

（A） 批处理操作系统必须在响应时间内处理完一个任务

（B） 实时操作系统必须在规定时间内处理完来自外部的事件

（C） 分时操作系统必须在周转时间内处理完来自外部的事件

（D） 分时操作系统必须在调度时间内处理完来自外部的事件

2.设有两个进程A、B，各按以下顺序使用P,V 操作进行同步。

```
A 进程:                    B 进程:
a1→                        b1→
   P(s1)                      P(s2)
a2 →                       b2→
   P(s2)                      P(s1)
a3→                        b3→
   V(s2)                      V(s1)
a4→                         b4→
   V(s1)                      V(s2)
a5→                        b5→
```

试问在下列执行顺序中，哪种情况会发生死锁？___D___

（A） a1,a2,a3,a4...（B） b1,b2,b3,b4,b5...（C） a1,a2,b1,b2,a3,b3...（D） a1,b1,a2,b2,a3,b3...

3. 在内存管理中，内存利用率高且保护和共享容易的是___D___内存管理方式

（A） 分区管理 （B）分页管理 （C）分段管理 （D）段页式管理

4．操作系统中，很多事件会引起调度程序的运行，但下列事件中不一定引起操作系统调度程序运行是___C___。

（A）当前运行着的进程出错。（B）当前运行着的进程请求输入/输出。

（C）有新的进程进入就绪状态。（D）当前运行的进程时间片用完。

5．操作系统中调度算法是核心算法之一，下列关于调度算法的论述中正确的是：___C___。

（A）先来先服务调度算法对既对长作业有利也对段作业有利。

（B）时间片轮调度算法转只对长作业有利。

（C）实时调度算法也要考虑作业的长短问题。

（D）高响应比优先调度算法既有利于短作业又兼顾长作业的作业还实现了先来先服务。

6.操作系统中产生死锁的根本原因是___C___。

（A）资源分配不当和CPU太慢 （B）系统资源数量不足

（C）作业调度不当和进程推进顺序不当 （D）用户数太多和CPU太慢

7．内存管理中把作业地址空间中使用的逻辑地址转变为内存中的物理地址称为___C___。

（A）链接。（B）装入。（C）重定位。（D）虚拟化。

8．I/O设备管理是操作系统的重要功能，那么下列对设备属性的描述正确的是___C___。

（A）字符设备的基本特征是可寻址到字节，即能指定输入的源地址或输出的目标地址。

（B）共享设备必须是可寻址的和可随机访问的设备。

（C）共享设备是指同一时间内运行多个进程同时访问的设备。

（D）在分配共享设备和独占设备时都可能引起进程死锁。

9．程序设计时需要调用操作系统提供的系统调用，被调用的系统调用命令经过编译后，形成若干参数和___A___

（A）访管指令或软中断 （B）启动I/O 指令 （C）屏蔽中断指令（D） 通道指令

10.以时间换空间或者以空间换时间是操作系统的基本技术，以下以空间换时间的机制是___C___。

（A）SPOOLING （B） 虚拟存储技术 （C） 通道技术 （D） 覆盖技术

填空题

(2018).（2分x7=14分）

1）操作系统的两大特征是（并发和共享 ）。

2）单道系统中，假设一批作业同时到达，若想平均周转时间最短，采用（短作业优先）调度算法。

3）时间片轮转调度算法中，如果时间片无穷大，该算法变成了（先来先服务）调度算法。

4）在某系统中有5个并发进程，都需要同类资源6个，问该系统肯定不会发生死锁时最少资源数是（26）。

5）对于首次适应算法、最佳适应算法和循环首次适应算法，可以保留高地址部分的大空闲区的算法是（首次适应算法）。

6）有m个进程共享同一临界资源，若使用信号量机制实现对一临界资源的互斥访问，则信号量的变化范围是（0-1）。

7）装入时动态链接和运行时动态链接这两种方式，（运行时动态链接）更节约内存。

简答题

(2013). 为什么要引入线程，线程和进程有何区别？（5分）

(2017). 系统型线程和用户型线程有何区别？（4分）

(2014). 从操作系统设计角度谈谈进程控制块的作用。（5分）

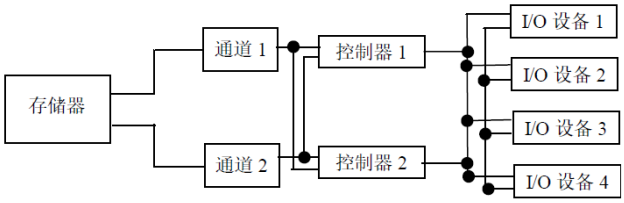
(2014). 静态链接和动态链接是现代操作系统中两种重要的链接方式，试比较同一程序经过静态链接和动态链接后的可执行文件大小，如果有不同分析原因。（5分）

(2017). 分段式系统和分页式系统有何区别？（4分）

(2016). 试说明页面置换算法在虚拟存储管理中的重要性。（2分）

(2016). FIFO算法适用于什么场合，又有何缺点。（2分）

(2013). 什么是通道，通道经常采用如图所示的交叉连接，为什么？（5分）



在CPU和设备控制器之间设置的，使一些原来由CPU处理的I/O任务转由通道来承担，从而把CPU从繁杂的I/O任务中解脱出来。上图为多通路方式，增加了设备到主机间的通路而不增加通道，不仅解决了“瓶颈”问题，而且提高了系统的可靠性，因为个别通道或控制器的故障不会使设备和存储器之间没有通路。

(2013). 简述操作系统引入缓冲的原因？（5分）

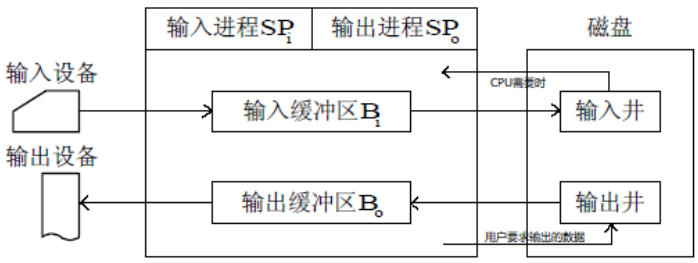
(1) 缓和CPU与I/O设备间速度不匹配的矛盾。(2) 减少对CPU的中断频率，放宽对CPU中断响应时间的限制。(3) 提高CPU和I/O设备之间的并行性。

(2017). 引入缓冲的目的是什么，有哪些常见的缓冲模式？（4分）

(1) 缓和CPU与I/O设备间速度不匹配的矛盾。(2) 减少对CPU的中断频率，放宽对CPU中断响应时间的限制。(3) 提高CPU和I/O设备之间的并行性。

单缓冲、双缓冲、循环缓冲、缓冲池。

(2017). SPOOLING技术如何实现，在操作系统中起何作用？（4分）



实现：首先保证建立在具有多道程序功能的操作系统上，而且还应有高速随机外存的支持；然后设计三个组成部分：（1）磁盘上开辟输入井和输出井，输入井是模拟脱机输入时的磁盘设备，用于暂存I/O设备输入的数据；输出井是模拟脱机输出时的磁盘，用于暂存用户程序的输出数据。（2）内存中开辟输入缓冲区和输出缓冲区，输入缓冲区用于暂存由输入设备送来的数据，以后再传送到输入井。输出缓冲区用于暂存从输出井送来的数据，以后再传送给输出设备。（3）输入进程和输出进程，输入进程模拟脱机输入时的外围控制机，将用户要求的数据从输入设备通过输入缓冲区再送到输入井，当CPU需要输入数据时，直接从输入井读入内存；输出进程模拟脱机输出时的外围控制机，把用户要求输出的数据先从内存送到输出井，待输出设备空闲时，再将输出井中的数据经过输出缓冲区送到输出设备上。

作用：(1) 提高了I/O的速度。(2) 将独占设备改造为共享设备。(3) 实现了虚拟设备功能。

(2013). 何谓文件的物理结构，可分为哪几类，有何优缺点？（5分）

文件的物理结构，又称为文件的存储结构，是指文件在外存上的存储组织形式。

连续分配，优点：(1) 顺序访问容易(2) 顺序访问速度快。缺点：(1) 要求有连续的存储空间(2) 必须事先知道文件的长度

链接分配，优点：由于链接分配是采取离散分配方式，消除了外部碎片，故而显著地提高了外存空间的利用率；又因为是根据文件的当前需要，为它分配必需的盘块，当文件动态增长时，可动态地再为它分配盘块，故而无需事先知道文件的大小。此外，对文件的增、删、改也十分方便。缺点：只适合于顺序访问，它对随机访问是极其低效的。此外，只通过链接指针来将一大批离散的盘块链接起来，其可靠性较差，因为只要其中的任何一个指针出现问题，都会导致整个链的断开。

计算题

1 大纲1.操作系统概述

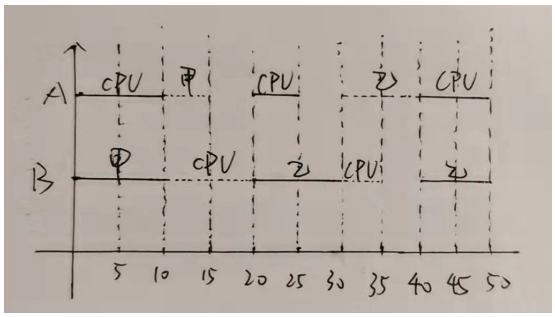
(2016.7). （10分）A、B 两个程序，程序A按顺序使用CPU10秒，使用设备甲5秒，使用CPU5秒，使用设备乙10秒，最后使用CPU10秒，程序B按顺序使用设备甲10秒，使用CPU10秒，使用设备乙10秒，使用CPU5秒，使用设备乙10秒。试问：

（1）在顺序环境下执行程序A和程序B，CPU的利用率是多少？（3分）

（2）在多道程序环境下，CPU的利用率是多少？请画出A、B程序的执行过程。（4分）

（3）多道批处理中，是否系统中并发的进程越多，资源利用率越好，为什么？（3分）

解答. （1）CPU运行时间为10+5+10+10+5=40s，两个程序运行的总时间为40+45=95s，利用率为40/95=42.1%

（2） 

CPU运行时间40s，两个程序运行总时间为50s，因此利用率为40/50=80%

（3）不是，若系统中进程过多，频繁进行切换，会导致性能下降，反而增大开销，降低利用率

注. 以后遇到此类题目，即给出几个不同的程序，每个程序以各个任务时间片给出时，一定要用甘特图来求解，因为其直观、快捷。为节省读者研究甘特图画法的时间，下面给出既定的步骤，读者可按下列步骤快速、正确地画出甘特图。

1. 横坐标上标出合适的时间间隔，纵坐标上的点是程序的名字。
2. 过横坐标上每个标出的时间点，向上作垂直于横坐标的虚线。
3. 用几种不同的线（推荐用“直线”“波浪线”“虚线”三种，较易区分）代表对不同资源的占用，按照题目给出的任务时间片，平行于横坐标把不同程序对应的线段分别画出来。

画图时要注意，如处理器、打印设备等资源是不能让两个程序同时使用的，有一个程序正在使用时，其他程序的请求只能排队。

2 大纲2.进程管理

2.1 进程同步——信号量和PV操作

最初由Dijkstra 把整型信号量定义为一个用于表示资源数目的整型量S，它与一般整型量不同，除初始化外，仅能通过两个标准的原子操作(Atomic Operation) wait(S)和signal(S)来访问。很长时间以来，这两个操作一直被分别称为P、V操作。

(2016.10). （10分）某工厂有两个生产车间和一个装配车间，生产车间生产A、B两种零件，装配车间把这两种零件装配成产品。生产车间甲把生产的A零件放到货架F1上，生产车间乙把生产的B零件放到货架F2上，假设两个货架的容量都是10个零件。装配车间每次从货架上取出一个A和一个B然后进行装配，请用P、V操作来进行正确的三个车间管理。

解答. 两个生产者一个消费者，这边一般把缓冲区（货架）视为临界资源。设置六个信号量：empty1和empty2分别表示货架F1和F2的空余容量，初值都设为10；full1和full2分别表示货架F1和F2的零件数量，初值都设为0；mutex1和mutex2分别用于货架F1和F2的独占操作。

生产车间甲：

```
while True:
    # 生产A零件
    P(empty1)  # 请求货架F1的一个空位置
    P(mutex1)  # 申请操作货架F1
    # 把生产的A零件放到货架F1上
    V(mutex1)  # 释放货架F1
    V(full1)  # 货架F1的A零件数量加一
```

生产车间乙：

```
while True:
    # 生产B零件
    P(empty2)  # 请求货架F2的一个空位置
    P(mutex2)  # 申请操作货架F2
    # 把生产的B零件放到货架F2上
    V(mutex2)  # 释放货架F2
    V(full2)  # 货架F2的B零件数量加一
```

装配车间：

```
while True:
    P(full1)  # 请求货架F1的A零件
    P(mutex1)  # 申请操作货架F1
    # 从货架上取一个A零件
    V(mutex1)  # 释放货架F1
    V(empty1)  # 货架F1的空闲空间数加一
    P(full2)  # 请求货架F2的B零件
    P(mutex2)  # 申请操作货架F2
    # 从货架上取一个B零件
    V(mutex2)  # 释放货架F2
    V(empty2)  # 货架F2的空闲空间数加一
    # 装配取出的A和B零件
```

注. P操作理解成请求资源(加锁)，V操作理解成释放资源(解锁)。

PV操作问题先判断类型是三种里的哪一种：生产者-消费者、读者-写者、哲学家进餐。然后根据相应的类型去设置合适的信号量，弄清楚什么资源是互斥的（信号量设为1），什么资源是用于同步的（信号量设为n）。生产者-消费者一般将某个缓冲池中的空余容量和满的容量设为信号量，有时候指明缓冲池是临界资源，还需单独设置一个互斥信号量。

(2013.2). （10分）假设有个南北向的胡同很窄，仅能容同方向的人顺序走过，相对方向的两个人则无法通过。现在胡同南北入口都有过路人。现把每个过路人当成一个进程，用P,V操作实现管理。

解答. 类读者-写者问题，相对方向的人互斥，同方向的人不存在互斥问题。设置整形变量tosourth和tonorth分别代表胡同中往南走和往北走的人数，初始值都设为0。设置三个初值都为1的互斥信号量，mutex1和mutex2用来实现对tosourth和tonorth的互斥访问，mutex用来实现对胡同的互斥使用。

南入口过路人：

```
P(mutex2)  # 防止同一方向的很多人一起过胡同，但是计数器显示人数为1的情况
tonorth = tonorth + 1  # 胡同内往北方向人数加一
if tonorth == 1:  # 如果是往北方向第一人
    P(mutex)  # 申请独占胡同，只允许同方向的人走
V(mutex2)  # 释放计数器互斥锁，让同方向的人可以申请
# 过胡同
P(mutex2)  # 过完胡同需要减少往北方向的人数，因此请求计数器互斥锁
tonorth = tonorth - 1  # 胡同内往北方向人数减一
if tonorth == 0:  # 如果是最后一个往北方向的人
    V(mutex)  # 申请释放胡同，让对面的人可以过
V(mutex2)  # 释放计数器互斥锁，让同方向的人可以申请
```

北入口过路人：

```
P(mutex1)
tosourth = tosourth + 1
if tosourth == 1:
    P(mutex)
V(mutex1)
# 过胡同
P(mutex1)
tosourth = tosourth - 1
if tosourth == 0:
    V(mutex)
V(mutex1)
```

(2015.2). （10分）某机场只有一条飞机跑道，为了提高效率和安全性，现规定：当飞机跑道有飞机起飞时，不允许飞机降落，但此时可以让多架飞机逐个利用跑道起飞；反之，当有飞机降落进入跑道时则不允许起飞飞机进入跑道，但允许飞机依次降落在跑道上，然后驶出跑道。请解决以下问题：（1）请利用信号量和P、V操作正确实现飞机在跑道上起降。（要求：说明所设的信号量的意义及初值）；（2）若把飞机看作进程，为了合理实现对飞机进程的管理，给出描述飞机进程的数据结构。

解答. （1）设置up,down分别为两个整形计数变量，代表跑道上起飞的飞机数和降落的飞机数，初值均为0。设置三个初值均为1的互斥信号量，mutex1和mutex2分别保证对up和down的互斥访问，mutex保证对跑道的互斥访问。

起飞的飞机：

```
P(mutex1)
up = up + 1
if (up == 1):
    P(mutex)
V(mutex1)
# 飞机起飞
P(mutex1)
up = up - 1
if (up == 0):
    V(mutex)
V(mutex1)
```

降落的飞机：

```
P(mutex2)
down = down + 1
if (down == 1):
    P(mutex)
V(mutex2)
# 飞机降落并驶出跑道
P(mutex2)
down = down - 1
if (down == 0):
    V(mutex)
V(mutex2)
```

（2）飞机的外部标识符——航班号；飞机的内部标识符——pid；飞机的状态——待起飞、起飞中、起飞完、待降落、降落中、降落完；飞机的优先级；飞机和塔台通信的上下文环境

注. 进程的数据结构用进程控制块（PCB）表示

(2018.15). （9分）某教学楼楼梯较窄，为了安全规定课间，一旦有人从上往下走，则不允许任何人从下往上走，但此时可以允许多人同时往下走，反之亦然。请设置合适的信号量（2分），应用PV操作完成此同步问题（5分），并分析是否会产生饥饿现象（2分）

解答.

注. 某一种作业长时间得不到机会运行的现象叫做“饥饿”现象

(2017.10). （9分）一家四口人，儿子喜欢吃苹果，由父亲负责购买，女儿喜欢吃橘子，由母亲负责购买。父亲和母亲购买水果后放到家中的抽屉里，儿子和女儿从抽屉里取出水果。假设抽屉只能容纳20个水果，同时只能一人开关，用记录型信号量同步父母子女四个进程。

解答. 设置记录型信号量empty代表抽屉剩余空间，初值为20，对应的链表指针为L；记录型信号量apple和orange分别代表抽屉内苹果和橘子数量，初值均为0，对应的链表指针为L；互斥信号量mutex用于控制抽屉的独占，初值为1。

```
def P(empty):
    empty.value = empty.value - 1
    if empty.value < 0:
        block(empty.L)

def P(apple):
    apple.value = apple.value - 1
    if apple.value < 0:
        block(apple.L)

def P(orange):
    orange.value = orange.value - 1
    if orange.value < 0:
        block(orange.L)

def V(empty):
    empty.value = empty.value + 1
    if empty.value <= 0:
        wakeup(empty.L)

def V(apple):
    apple.value = apple.value + 1
    if apple.value <= 0:
        wakeup(apple.L)

def V(orange):
    orange.value = orange.value + 1
    if orange.value <= 0:
        wakeup(orange.L)
```

父亲：

```
while True:
    P(empty)  # 看看有没有空位置，没有的话就去主卧的empty.L床上睡觉，有的话继续下述步骤
    # 买一个苹果
    P(mutex)  # 打开抽屉
    # 放入买的那个苹果
    V(mutex)  # 关上抽屉
    V(apple)  # 苹果数量加一，如果之前是儿子想吃苹果但发现没有便去睡觉了，则唤醒apple.L床上的儿子
```

儿子：

```
while True:
    P(apple)  # 看看有没有苹果吃，没有的话就去自己卧室的apple.L床上睡觉，等父亲放好苹果叫醒自己
    # 多谢父亲
    P(mutex)  # 打开抽屉
    # 吃一个苹果
    V(mutex)  # 关上抽屉
    V(empty)  # 抽屉多一个空位，如果之前父亲或母亲看抽屉没有空位置就去主卧睡觉了，叫醒他（她）
```

母亲：

```
while True:
    P(empty)
    # 买一个橘子
    P(mutex)
    # 放入买的那个橘子
    V(mutex)  # 关上抽屉
    V(orange)  # 橘子数量加一，如果之前是女儿想吃橘子但发现没有便去睡觉了，则唤醒orange.L床上的女儿
```

女儿：

```
while True:
    P(orange)  # 看看有没有橘子吃，没有的话就去自己卧室的orange.L床上睡觉，等母亲放好橘子叫醒自己
    # 母亲辛苦了
    P(mutex)  # 打开抽屉
    # 吃一个橘子
    V(mutex)  # 关上抽屉
    V(empty)  # 抽屉多一个空位，如果之前父亲或母亲看抽屉没有空位置就去主卧睡觉了，叫醒他（她）
```

注. 在整型信号量机制中的wait操作，只要是信号量S≤0，就会不断地测试。因此，该机制并未遵循“让权等待”的准则，而是使进程处于“忙等”的状态。记录型信号量机制则是一种不存在“忙等”现象的进程同步机制。但在采取了“让权等待”的策略后，又会出现多个进程等待访问同一临界资源的情况。为此，在信号量机制中，除了需要一个用于代表资源数目的整型变量value外，还应增加一个进程链表指针L，用于链接上述的所有等待进程。

进程被唤醒后是从之前挂起的位置开始往下运行，因此挂起时父亲或母亲把empty变为-1，或是儿子把apple变为-1，在被唤醒后虽然empty和apple的值都是0，但可以直接放水果或是吃苹果，不必再让empty或者apple减1了。

不同进程中对同一个值为1的信号量既有P也有V，则是互斥信号量（本题的抽屉）；对同一个记录型信号量的P和V操作分别分散在不同的进程中，则实现了等待和唤醒的功能（消息缓冲通信中的资源信号量sm和本题的empty、apple、orange）；对同一个整型信号量的P和V操作分别分散在不同的进程中，则实现了同步的功能（生产者-消费者中的empty、full）

2.2 进程通信——消息缓冲队列通信机制

消息缓冲队列通信机制首先由美国的Hansan提出，并在RC 4000系统上实现，后来被广泛应用于本地进程之间的通信中。在这种通信机制中，发送进程利用Send原语将消息直接发送给接收进程；接收进程则利用Receive原语接收消息。

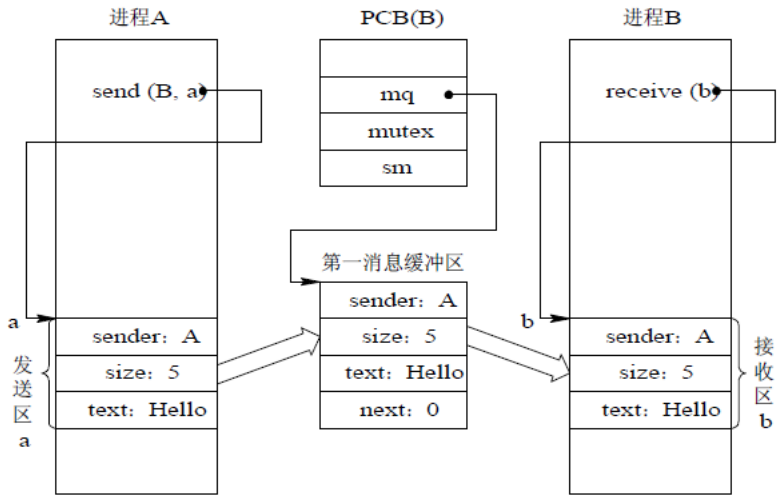


图 2-14 消息缓冲通信

(2014.3). （6分）简述消息缓冲队列通信机制，并用信号量和wait，signal操作实现消息缓冲队列通信机制中的发送和接收原语。

解答. 发送进程利用Send原语将消息直接发送给接收进程；接收进程则利用Receive原语接收消息。

发送原语：

```
def Send(receiver, a):
    i = getbuf(a.size)  # 根据a.size申请缓冲区i
    i.sender = a.sender  # 将发送区a中的信息复制到缓冲区i中
    i.size = a.size  # 消息长度
    i.text = a.text  # 消息正文
    i._next = 0  # 指向下一个消息缓冲区的指针
    j = getid(PCBset, receiver)  # 获得接收进程内部标识符
    wait(j.mutex)  # 占用j的互斥信号量
    insert(j.mq, i)  # 将消息缓冲区插入消息队列
    signal(j.mutex)  # 释放互斥信号量
    signal(j.sm)  # 唤醒资源信号量上的j
```

接收原语：

```
def reveive(b):
    j = internal name  # j为接收进程的内部标识符
    wait(j.sm)  # 等待接收消息通知
    wait(j.mutex)  # 消息来了以后占用自己的互斥信号量
    i = pop(j.mq)  # 取出消息队列中的第一条消息
    signal(j.mutex)  # 释放互斥信号量
    b.sender = i.sender  # 将消息缓冲区i中的信息复制到接收区b
    b.size = i.size  # 消息长度
    b.text = i.text  # 消息正文
```

3 大纲3.调度与死锁

3.1 调度算法

在早期的时间片轮转法中，系统将所有的就绪进程按先来先服务的原则排成一个队列，每次调度时，把CPU分配给队首进程，并令其执行一个时间片。时间片的大小从几ms 到几百ms。当执行的时间片用完时，由一个计时器发出时钟中断请求，调度程序便据此信号来停止该进程的执行，并将它送往就绪队列的末尾；然后，再把处理机分配给就绪队列中新的队首进程，同时也让它执行一个时间片。

短作业优先(SJF)的调度算法是从后备队列中选择一个或若干个估计运行时间最短的作业，将它们调入内存运行。而短进程优先(SPF)调度算法则是从就绪队列中选出一个估计运行时间最短的进程，将处理机分配给它，使它立即执行并一直执行到完成，或发生某事件而被阻塞放弃处理机时再重新调度。

抢占式优先权调度算法，在这种方式下，系统同样是把处理机分配给优先权最高的进程，使之执行。但在其执行期间，只要又出现了另一个其优先权更高的进程，进程调度程序就立即停止当前进程(原优先权最高的进程)的执行，重新将处理机分配给新到的优先权最高的进程。因此，在采用这种调度算法时，是每当系统中出现一个新的就绪进程i 时，就将其优先权Pi与正在执行的进程j 的优先权Pj进行比较。如果Pi≤Pj，原进程Pj便继续执行；但如果是Pi>Pj，则立即停止Pj的执行，做进程切换，使i进程投入执行。

(2016.8).（10分）考虑5个进程P1、P2、P3、P4、P5，如下表，规定进程的优先级越小，优先级越高，试计算在采用下述几种调度算法时各个进程周转时间和带权周转时间。假设忽略进程的调度时间。

（1）先来先服务调度算法（FCFS）；

（2）时间片轮转调度算法（时间片为1ms）（RR）；

（3）最短作业优先调度算法（SJF）；

（4）剥夺式优先级调度算法（HPF）。

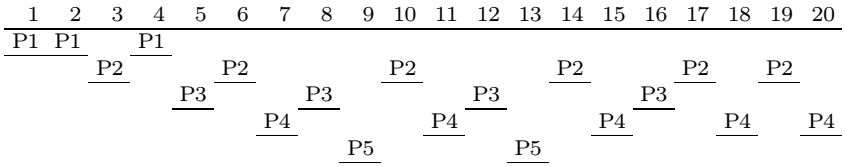
进程	提交时刻	需要的CPU时间(ms)	优先级
P1	0	3	3
P2	2	6	5
P3	4	4	1
P4	6	5	2
P5	8	2	4

解答.

（1）

进程	提交时刻	需要的CPU时间(ms)	开始执行时间	完成时间	周转时间	带权周转时间
P1	0	3	0	3	3	1
P2	2	6	3	9	7	7/6
P3	4	4	9	13	9	2.25
P4	6	5	13	18	12	2.4
P5	8	2	18	20	12	6

（2）



进程	提交时刻	需要的CPU时间(ms)	完成时间	周转时间	带权周转时间
P1	0	3	4	4	4/3
P2	2	6	19	17	17/6
P3	4	4	16	12	3
P4	6	5	20	14	2.8
P5	8	2	13	5	2.5

（3）

进程	提交时刻	需要的CPU时间(ms)	完成时间	周转时间	带权周转时间
P1	0	3	3	3	1
P2	2	6	9	7	7/6
P3	4	4	15	11	2.75
P4	6	5	20	14	2.8
P5	8	2	11	3	1.5

（4）

进程	提交时刻	需要的CPU时间(ms)	完成时间	周转时间	带权周转时间
P1	0	3	11	11	11/3
P2	2	6	8	6	1
P3	4	4	20	16	4
P4	6	5	16	10	2
P5	8	2	10	2	1

注. 周转时间=完成时间-到达时间（提交时刻）

作业的周转时间T与系统为它提供服务的时间Ts（需要的CPU时间）之比，即W = T/Ts，称为带权周转时间

时间片轮转调度算法中执行完当前时间片会把当前进程送往队列末尾，在下一个时间片执行队首的进程之前会先判断有没有新的进程到来，如果有的话将新的进程插入队首，因此每个进程在它的提交时刻对应的时间片必然是该进程在运行

最短作业优先调度算法在选了作业之后是让它一直执行到完成的，所以正确的流程是先执行最先到达的作业，在执行完之后看一下还到了哪些作业，从中选出一个需要时间最短的

(2017.7). （7分）设有三道作业，它们的提交时间及执行时间由下表给出：

作业号	提交时间	执行时间
1	8.5	2.0
2	9.2	1.6
3	9.4	0.5

试计算在单道程序环境下，采用先来先服务调度算法和最短作业优先调度算法时的平均周转时间。

解答. 先来先服务调度算法：

作业号	提交时间	执行时间	开始时间	完成时间	周转时间
1	8.5	2.0	8.5	10.5	2.0
2	9.2	1.6	10.5	12.1	2.9
3	9.4	0.5	12.1	12.6	3.2

平均周转时间=(2.0+2.9+3.2)/3=2.7

最短作业优先调度算法：

作业号	提交时间	执行时间	开始时间	完成时间	周转时间
1	8.5	2.0	8.5	10.5	2.0
2	9.2	1.6	11	12.6	3.4
3	9.4	0.5	10.5	11	1.6

平均周转时间=(2.0+3.4+1.6)/3=7/3≈2.33

注. 平均周转时间$T=\frac{1}{n}\left[\sum_{i=1}^{n}T_i\right]$，也就是各周转时间加起来除以总作业（进程）数

平均带权周转时间$W=\frac{1}{n}\left[\sum_{i=1}^{n}\frac{T_i}{T_s}\right]$，也就是各带权周转时间加起来除以总作业（进程）数

(2018.12). （9分）多道程序系统有一个CPU和两台独占设备，即IO1和IO2，现在有3个优先级别从高到低的作业J1、J2、J3 到达，它们使用资源的先后顺序和占用时间分别是：

J1：IO2（60）；CPU（20）；IO1（60）；CPU（20）

J2：IO1（40）；CPU（40）；IO2（80）

J3：CPU（60）；IO1（40）

假设处理机调度采用可抢占的优先级算法，设备不能抢占，忽略调度时间，时间单位为分钟。计算下列问题：

1）分别计算3个作业的周转时间（3 分）

2）3个作业全部完成时CPU的利用率（3 分）

3）3个作业全部完成时IO1的利用率（3 分）

3.2 预防死锁的方法——银行家算法

所谓安全状态，是指系统能按某种进程顺序(P1，P2，...，Pn)(称〈P1，P2，...，Pn〉序列为安全序列)，来为每个进程Pi分配其所需资源，直至满足每个进程对资源的最大需求，使每个进程都可顺利地完成。如果系统无法找到这样一个安全序列，则称系统处于不安全状态。

(2014.4). （6分）设系统种有三种类型的资源（A,B,C）和五个进程（P1,P2,P3,P4,P5），A的资源的数量为17，B的资源的数量为5，C的资源的数量为20，在T0时刻状态如下：

	最大资源需求量			已分配资源需求量		
	A	B	C	A	B	C
P1	5	5	9	2	1	2
P2	5	3	6	4	0	2
P3	4	0	11	4	0	5
P4	4	2	5	2	0	4
P5	4	2	4	3	1	4

剩余资源数	A	B	C
	2	3	3

系统采用银行家算法实施死锁避免策略。

（1）T0时刻是否为安全状态？若是请给出安全序列。

（2）在T0时刻，若进程P2请求资源（0，3，4），是否能实施资源分配？为什么？

（3）在（2）的基础上，若进程P4请求资源（2，0，1），是否能实施资源分配？为什么？

解答. （1）写出需求矩阵

	尚需资源		
	A	B	C
P1	3	4	7
P2	1	3	4
P3	0	0	6
P4	2	2	1
P5	1	1	0

，利用安全性算法对T0时刻的资源分配情况进行分析

	Work			Need			Allocation			Work+Allocation			Finish
	A	B	C	A	B	C	A	B	C	A	B	C	
P5	2	3	3	1	1	0	3	1	4	5	4	7	true
P4	5	4	7	2	2	1	2	0	4	7	4	11	true
P1	7	4	11	3	4	7	2	1	2	9	5	13	true
P2	9	5	13	1	3	4	4	0	2	13	5	15	true
P3	13	5	15	0	0	6	4	0	5	17	5	20	true

可得到一个安全序列〈P5，P4，P1，P2，P3〉，故T0时刻是安全状态。

（2）不能，因为C的剩余资源数只有3，不能满足进程P2的请求。

（3）能，系统按银行家算法进行检查：①$\text{Request}_4(2,0,1) \leqslant \text{Need}_4(2,2,1)$ ②$\text{Request}_4(2,0,1) \leqslant \text{Available}(2,3,3)$ ③系统先试探性地为P4分配资源，并修改Availalbe, Allocation_4和Need_1向量，然后利用安全性算法分析，

	Work			Need			Allocation			Work+Allocation			Finish
	A	B	C	A	B	C	A	B	C	A	B	C	
P4	0	3	2	0	2	0	4	0	5	4	3	7	true
P2	4	3	7	1	3	4	4	0	2	8	3	9	true
P3	8	3	9	0	0	6	4	0	5	12	3	14	true
P5	12	3	14	1	1	0	3	1	4	15	4	18	true
P1	15	4	18	3	4	7	2	1	2	17	5	20	true

可以找到一个安全序列〈P4，P2，P3，P5，P1〉。

因此系统是安全的，可以实施资源分配。

注. 判断是否为安全状态需要用安全性算法，该算法在做本题时的步骤如下：

1. 先写出需求矩阵Need，表示每一个进程尚需的各类资源数(用最大资源需求量矩阵减去已分配资源需求量矩阵)

2. 画出表格，从左到右每一列分别表示进程名(待定)、工作向量Work(表示系统可提供给进程继续运行所需的各类资源数目)、需求矩阵Need中对应的该行进程的值(第一列的进程名定出来之后再写)、分配矩阵Allocation中对应的该行进程的值(第一列的进程名定出来之后再写)、Work+Allocation(表示该行进程运行完毕释放资源后系统可提供给进程继续运行所需的各类资源数目)、Finish向量(表示系统是否有足够的资源分配给该行进程)

3. 首先在工作向量Work第一行写上当前剩余资源数(2 3 3)，然后查看需求矩阵Need，发现P4和P5都符合资源分配条件，可以随便选一个（不会出现选其中一个得到不安全状态，而选另一个得到安全状态的情况，因为选了其中一个之后运行完毕资源释放时至少会把从当前剩余资源数请求的资源给分配出去，而这个资源量又是够另一个运行的，所以如果有多个进程都符合资源分配条件，那么这些进程都可以运行完，而与顺序无关），这里我们选择P5，把P5填入(2 3 3)左边，P5对应的Need和Allocation矩阵中的值(1 1 0)和(3 1 4)分别填入对应位置。P5运行完毕将释放资源，用Work+Allocation计算此时的剩余资源数（因为是先从Work申请资源，然后再把自己占用的所有资源全部释放，所以剩余资源数=(Work-Need)+Max=Work+(Max-Need)=Work+Allocation），得到(5 4 7)，并将true填入最后一列，代表该进程可以分配足够的资源使之运行完成。

4. 把第一行的(5 4 7)填入第二行Work处，然后继续第3步的操作，已经标记为true的进程不需要再去需求矩阵中找，（这边因为上一步的P4和P5都符合条件，优先把都符合条件的先处理掉）

5. 对表格进行检查，保证满足三点：①上一行的Work+Allocation抄写到下一行的Work没有抄错 ②Work中的每一行都要大于等于Need ③Work+Allocation最后一行的值等于各资源的数量

判断能否实施资源分配用银行家算法，有三个要点：

1. 首先看请求资源是否不超过尚需资源

2. 然后看请求资源是否不超过剩余资源

3. 最后看资源分配之后系统是否处于安全状态

(2017.8). （9分）某系统有A、B、C、D四类资源可供五个进程P1、P2、P3、P4、P5共享。系统共有这四类资源为：A类3个、B类14个、C类12个、D类12个。进程对资源的需求和分配情况如下：

进程	已占有资源				最大需求数			
	A	B	C	D	A	B	C	D
P1	0	0	1	2	0	0	1	2
P2	1	0	0	0	1	7	5	0
P3	1	3	5	4	2	3	5	6
P4	0	6	3	2	0	6	5	2
P5	0	0	1	4	0	6	5	6

（1）现在系统是否处于安全状态？（4分）

（2）如果进程P2提出需要A类资源0个、B类资源4个、C类资源2个和D类资源0个，系统能否去满足它的请求？（5分）

解答. （1）写出需求矩阵

进程	尚需资源			
	A	B	C	D
P1	0	0	0	0
P2	0	7	5	0
P3	1	0	0	2
P4	0	0	2	0
P5	0	6	4	2

，根据安全性算法对系统现在的情况进行分析，

进程	Work				Need				Allocation				Work+Allocation				Finish
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	
P1	1	5	2	0	0	0	0	0	0	0	1	2	1	5	3	2	true
P4	1	5	3	2	0	0	2	0	0	6	3	2	1	11	6	4	true
P2	1	11	6	4	0	7	5	0	1	0	0	0	2	11	6	4	true
P3	2	11	6	4	1	0	0	2	1	3	5	4	3	14	11	8	true
P5	3	14	11	8	0	6	4	2	0	0	1	4	3	14	12	12	true

可得到一个安全序列〈P1，P4，P2，P3，P5〉，故现在系统处于安全状态。

（2）系统按银行家算法进行检查：①$\text{Request}_2(0,4,2,0) \leqslant \text{Need}_2(0,7,5,0)$ ②$\text{Request}_2(0,4,2,0) \leqslant \text{Availalbe}(1,5,2,0)$ ③系统先试探性地为进程P2分配资源，并修改Available, Allocation_2和Need_2向量，然后利用安全性算法分析，

进程	Work				Need				Allocation				Work+Allocation				Finish
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	
P1	1	1	0	0	0	0	0	0	0	0	1	2	1	1	1	2	true
P3	1	1	1	2	1	0	0	2	1	3	5	4	2	4	6	6	true
P2	2	4	6	6	0	3	3	0	1	4	2	0	3	8	8	6	true
P4	3	8	8	6	0	0	2	0	0	6	3	2	3	14	11	8	true
P5	3	14	11	8	0	6	4	2	0	0	1	4	3	14	12	12	true

可以找到一个安全序列〈P1，P3，P2，P4，P5〉，因此系统是安全的，可以满足进程P2的请求。

(2018.14). （9分）五个进程P1，P2，P3，P4，P5均需要使用资源A、B、C。其中，A、B、C资源的总数分别为10，5，7。当前已分配资源情况和各进程的最大资源需求如下表所示。

进程	最大需求资源	已分配资源
P1	(7,5,3)	(0,1,0)
P2	(3,2,2)	(2,0,0)
P3	(9,0,2)	(3,0,2)
P4	(2,2,2)	(2,1,1)
P5	(4,3,3)	(0,0,2)

1）什么是安全状态？（2分）

2）系统进入不安全状态是否一定会产生死锁？（3分）

3）P1请求资源(1,0,2)，请根据银行家算法判断是否应该为其分配资源（4分）

3.3 死锁的检测与解除——资源分配图和死锁定理

资源分配图：我们用圆圈代表一个进程，用方框代表一类资源。由于一种类型的资源可能有多个，我们用方框中的一个点代表一类资源中的一个资源。此时，请求边是由进程指向方框中的rj，而分配边则应始于方框中的一个点。

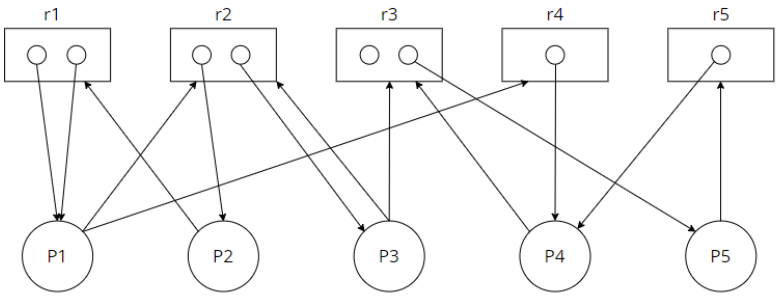
可以证明：S为死锁状态的充分条件是：当且仅当S状态的资源分配图是不可完全简化的。该充分条件被称为死锁定理。

(2015.6). （5分）假设系统有五类独占资源：r1，r2，r3，r4，r5，各类资源分别有：2，2，2，1，1个单位的资源，有五个进程：P1，P2，P3，P4，P5，其中P1已占有2个单位的r1，且申请一个单位的r2和一个单位的r4；P2已占有一个单位的r2，且申请一个单位的r1；P3已占有一个单位的r2且申请一个单位的r2和一个单位的r3；P4已占有一个单位的r4和一个单位的r5，且申请一个单位的r3；P5已占有一个单位的r3且申请一个单位的r5。

（1）试画出该时刻的资源分配图。（2分）

（2）什么是死锁定理，如何判断给出的资源分配图中有无死锁，给出判断过程和结果。（3分）

解答. （1）



（2）死锁定理：对于某个状态S，当且仅当S状态的资源分配图是不可完全简化的，则S为死锁状态。

1. 将一个单位的r3分配给P4，P4运行完毕后释放其占有的全部资源，消去P4所有的请求边和分配边
2. 将一个单位的r5分配给P5，P5运行完毕后释放其占有的全部资源，消去P5所有的请求边和分配边
3. 后续无法再消去任何请求边和分配边，因此图中有死锁

4 大纲4.内存管理

4.1 基本分页存储管理方式

(2013.4).（7分）进程某时刻的页表如下图所示：

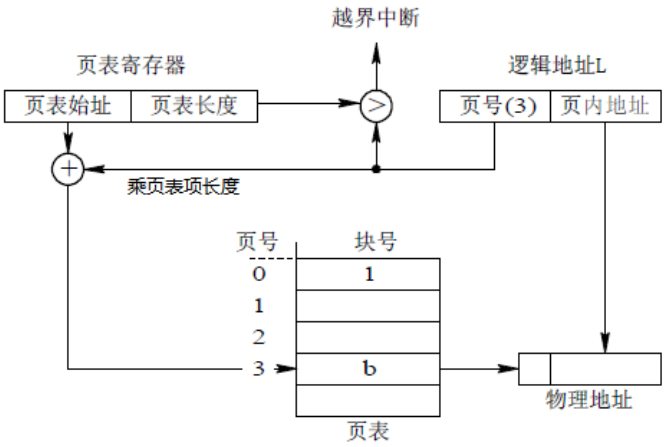
页号	标志	主存块号
0	1	4
1	1	8
2	0	
3	1	2
4	0	
5	1	0

其中的数字为十进制，页号、块号都以0开始，页的大小为2K字节，标志为1是在内存，标志为0表示不在内存。请回答下列问题：

（1）简述分页式虚拟存储系统中，一个逻辑地址到物理地址的转换过程（并画出地址转换机构图）。

（2）逻辑地址0x1830和0x206B对应的物理地址是什么？

解答.（1）从逻辑地址得到页号和页内地址。将页号与页表寄存器中的页表长度进行比较，如果页号大于等于页表长度，则产生地址越界中断；否则将页号乘上页表项长度后加上页表寄存器中的页表始地址，得到该页表项在实际的内存中的位置，于是可从中得到该页号对应的物理块号，将其装入物理地址寄存器中。同时，将页内地址装入物理地址寄存器的块内地址字段中。这样便完成了从逻辑地址到物理地址的转换。



（2）首先把十六进制转化为二进制，0x1830 => 0001 1000 0011 0000，0x206B => 0010 0000 0110 1011；页的大小为2K字节，则低11位为页内地址，其余部分为页号，两个页号分别为3(00011B)和4(00100B)，页号3对应的主存块号为2，因此逻辑地址0x1830对应的物理地址为0x1030(0001 0000 0011 0000)，而逻辑地址0x206B因为不在内存，所以没有物理地址。

注. 地址越界的判断中等于也越界是因为页号从0开始，当它等于页表长度的时候实际上就已经溢出一个页表项了。

手工计算物理地址的时候这么简单是因为省去了第（1）问中系统需要经过计算得到该页表项在实际的内存中的位置的过程，因为页表已经直接给出来了，我们直接肉眼观察即可。

(2014.5).（6分）一个进程某时刻的页表如下图所示：

页号	标志	内存块号
0	1	2
1	0	
2	1	8
3	1	1
4	0	
5	1	0

本题中的数字均为十进制，页号、块号都以0开始，页的大小为2K字节，标志为1表示页面在内存，标志为0表示不在内存；请回答下列问题：

（1）简述分页式虚拟存储系统中，一个逻辑地址到物理地址的转换过程（并画出地址转换机构图）

（2）逻辑地址5188和3199对应的物理地址是什么？

解答.（1）略（2）法一：5188对应的页号和页内地址分别为INT$\left[\frac{5188}{2048}\right]=2, 5188 \bmod 2048=1092$，根据页表知对应的内存块号为8，则十进制下对应的物理地址为$8 \times 2048+1092=17476$；3199对应的页号和页内地址分别为INT$\left[\frac{3199}{2048}\right]=1, 3199 \bmod 2048=1151$，根据页表知不在内存中，则不存在对应的物理地址。

法二：5188转化为二进制为1 0100 0100 0100，页的大小为2K字节，则低11位为页内地址，其余部分为页号，十进制的页号为2(10B)，根据页表知对应的内存块号为8，因此二进制下对应的物理地址为100 0100 0100 0100，转化成十进制为17476；3199转化为二进制为1100 0111 1111，页号为1(1B)，根据页表知不在内存中，故不存在对应的物理地址。

注．求逻辑地址对应的物理地址时可以转化成二进制，根据页的大小判断低位的位数对应页内地址，2的n次幂字节下这个n就是位数，然后逻辑地址的剩余高位数字代表页号，转成十进制后再去查表。也可以直接用十进制下的两个公式求页号和页内地址，这个数字直接代表了字节数，因此页的大小参与计算时要把K字节乘上1024。

(2017.9).（9分）某分页系统，每个页面长为1KB，某时刻该用户进程的页表如下：

页号	物理块号	是否在快表
0	8	是
1	7	是
2	4	否
3	10	否
4	5	否
5	3	是
6	2	是

（1）请写出分页系统的地址转换过程（3分）

（2）计算两个逻辑地址：0AC5H、1AC5H对应的物理地址（16进制表示）。（3分）

（3）已知主存的一次存取为2us，对于快表的查询时间可以忽略，则访问上述两个逻辑地址分别耗费多少时间？（3分）

解答.（1）逻辑地址除以页面大小后取整得到页号，逻辑地址对页面大小作取余运算得到页内地址。先将页号与页表寄存器中的页表长度进行比较，如果页号大于等于页表长度，则产生地址越界中断；否则将页号乘上页表项长度后加上页表寄存器中页表始地址，得到该页表项在实际的内存中的位置，于是可从中得到该页号对应的物理块号，将其装入物理地址寄存器中。同时，将页内地址装入物理地址寄存器的块内地址字段中。这样便完成了从逻辑地址到物理地址的转换。

（2）法一：首先将十六进制转换成十进制：0AC5H => 2757，1AC5H => 6853；然后分别用公式算出页号和页内地址，INT$\left[\frac{2757}{1024}\right]=2$，$2757 \bmod 1024=709$，INT$\left[\frac{6853}{1024}\right]=6$，$6853 \bmod 1024=709$；由页表知两个逻辑地址对应的物理块号分别为4和2，于是物理地址分别为$4 \times 1024+709=4805, 2 \times 1024+709=2757$，转换成十六进制分别是0x12C5和0x0AC5。

法二：将逻辑地址0AC5H转化为二进制0000 1010 1100 0101，每个页面长为1KB，则低10位为页内地址，剩余部分为页号2(10B)，查页表知对应的物理块号为4，于是二进制的物理地址为01 0010 1100 0101，转为十六进制0x12C5；将逻辑地址1AC5H转化为二进制0001 1010 1100 0101，则页号为6(110B)，查页表知对应的物理块号为2，于是二进制的物理地址为1010 1100 0101，转为十六进制0x0AC5。

（3）对于地址1AC5H，由于在快表中，可以直接获取物理地址，因此访问的时间就是一次取主存的时间2us；对于地址0AC5H，由于不在快表中，系统需要先从主存中的页表中获取物理块号然后才能访问，因此时间等于两次取主存的时间4us。

4.2 基本分段存储管理方式

(2015.3).（5分）某段式存储管理系统中采用如下段表：（用十进制）

段号	段的长度（字节）	主存起始地址
0	500	150
1	180	800
2	600	1000
3	1680	1850

试回答：

（1）计算[0, 150], [1, 98], [2, 601], [3, 50]的内存地址，其中方号内的第一元素为段号，第二元素为段内地址。

（2）存取主存中的一条指令或数据至少要访问几次内存？如何提高速度？

解答.（1）[0, 150]: 150+150=300

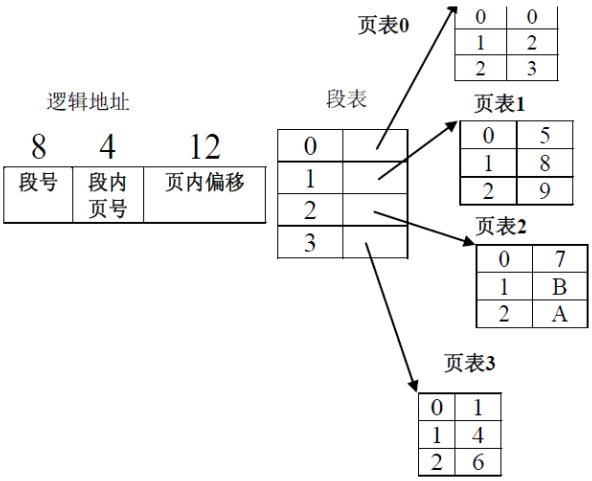
[1, 98]: 800+98=898

[2, 601]: 段内地址超过段长，发出越界中断信号

[3, 50]: 1850+50=1900

（2）至少访问两次，可以增设一个快表，用于保存最近常用的段表项。

(2016.9).（10分）某系统采用段页式存储管理，有关的数据结构如下图所示。



（1）说明在段页式系统中动态地址变换过程。（4分）

（2）计算虚地址200804（十进制）的物理地址（用十进制表示）。（3分）

（3）计算物理地址32784（十进制）的虚地址（用十进制表示）。（3分）

解答.（1）先将段号与段表寄存器中的段表长度进行比较，若段号小于段表长度，表示未越界，用段号乘上段表项长度再加上段表寄存器中的段表始址计算出该段表项在实际内存中的位置，从中得到该段的页表始址，利用逻辑地址中的段内页号来获得对应页的页表项在实际内存中的位置，从中读出该页所在的物理块号，利用块号和页内偏移最终计算出物理地址。

（2）首先把十进制200804转化为二进制0000 0011 0001 0000 0110 0100，则段号为3(11B)，段内页号为1(0001B)，页内地址为100(0000 0110 0100B)，页内偏移12位，说明页的大小为4KB，查段表和页表知物理块为4，于是物理地址为$4\times4096+100=16484$。

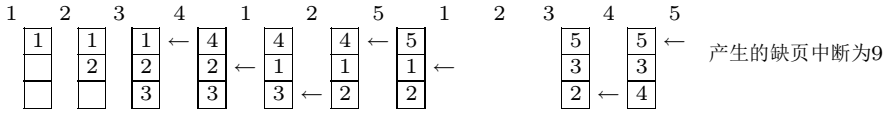
（3）物理块为$\mathrm{INT}\left[\frac{32784}{4096}\right]=8$，页内偏移$32784 \bmod 4096=16$(0000 0001 0000B)，观察上图知段号为1(0000 0001B)，页号为1(0001B)，拼出二进制下的逻辑地址0000 0001 0001 0000 0001 0000，转化为十进制69648。

4.3 页面置换算法

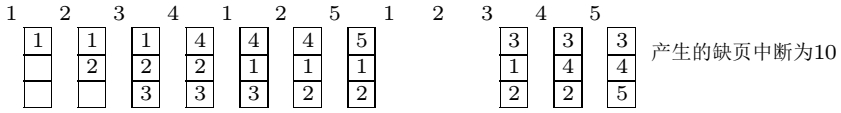
在进程运行过程中，若其所要访问的页面不在内存而需把它们调入内存，但内存已无空闲空间时，为了保证该进程能正常运行，系统必须从内存中调出一页程序或数据送磁盘的对换区中。但应将哪个页面调出，须根据一定的算法来确定。通常，把选择换出页面的算法称为页面置换算法(Page-Replacement Algorithms)。

(2016.6(3)).（6分）设页面走向为1,2,3,4,1,2,5,1,2,3,4,5，当物理页框数分别是3和4时，试问：采用FIFO、LRU置换算法产生的缺页中断分别是多少？（这里假设内存开始时都是空的并且只要是第一次用到的页面都产生缺页中断）

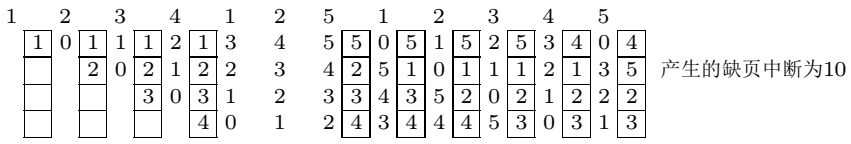
解答. 物理页框数为3时，FIFO置换算法：



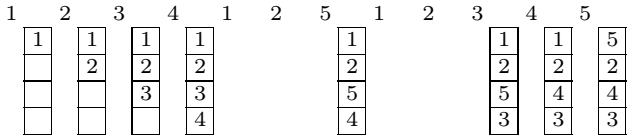
LRU置换算法：



物理页框数为4时，FIFO置换算法：



LRU置换算法：



注. 注意问的是缺页中断还是页面置换次数，前面把内存填满的过程不发生页面置换，但是都算缺页中断。

FIFO置换算法可以用一个箭头标识当前最久的页面，把所有的页面置换都挤压在一起时，箭头总是从上到下、再从上到下排布，如上面第一张图所示；也可以在每一次页面走向后都标上当前页面运行的时间，每次需要置换时选择时间最大的，然后在置换后的对应位置时间标记为0，如上面第三张图所示。

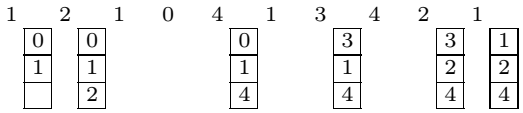
LRU置换算法在每次发生置换时都在第一行的页面走向的当前位置往前看，找到待置换的那些数字中离自己最远的那个进行置换。

(2014.1). （6分）某操作系统采用分页式虚拟存储管理办法，现有一个进程需要访问的地址序列（字节）分别是：115，228，120，88，446，102，321，432，260，167，假设该进程的第0页已经装入内存，并分配给该进程300字节内，页的大小为100字节，试回答以下问题：

（1）按LRU调度算法将产生多少次页面置换，依次淘汰的页号是什么？页面置换率为多少？

（2）LRU页面置换算法的基本思想是什么？

解答. （1）将地址序列转换为页面：1，2，1，0，4，1，3，4，2，1

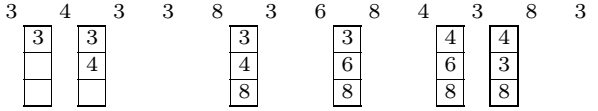


产生4次页面置换，依次淘汰页号2、0、1、3，页面置换率为4/10=40%

（2）由于无法预测各页面将来的使用情况，只能利用“最近的过去”作为“最近的将来”的近似，认为最近最久未使用的页面在最近的将来也不太可能用到。因此，LRU 置换算法是选择最近最久未使用的页面予以淘汰。

(2015.4(1)). （5分）LRU的思想和依据是什么？请利用LRU算法解决下列问题：在一个请求分页系统中，假如系统分配给一个作业的物理块数为3，此作业的页面走向为3，4，3，3，8，3，6，8，4，3，8，3。试用LRU算法计算页面置换次数。

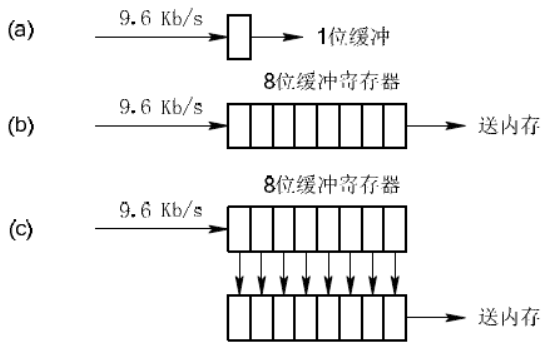
解答. 依据：由于无法预测各页面将来的使用情况，只能利用“最近的过去”作为“最近的将来”的近似，认为最近最久未使用的页面在最近的将来也不太可能用到。思想：因此，LRU 置换算法是选择最近最久未使用的页面予以淘汰。



5 大纲5.设备管理

5.1 缓冲管理

(2015.5). （5分）按照下图说明操作系统中引入缓冲的好处



解答. 如果发来的数据仅用一位缓冲来接收，如图（a）所示，则必须在每收到一位数据时便中断一次CPU，这样，对于速率为9.6Kb/s的数据通信来说，就意味着其中断CPU的频率也为9.6Kb/s，每0.0001秒就要中断CPU一次，而且CPU必须在0.0001秒内予以响应，否则缓冲区内的数据将被冲掉。倘若设置一个具有8位的缓冲寄存器，如图（b）所示，则可使CPU被中断的频率降低位原来的1/8；若再设置一个8位寄存器，如图（c）所示，则又可把CPU对中断的响应时间放宽到0.0008秒。因此引入缓冲的其中一部分好处就在于可以减少CPU的中断频率，放宽对CPU中断响应时间的限制。

注.（b）体现了减少中断频率，是因为可以等到8位缓冲装满以后再一口气发给CPU；（b）体现了放宽对中断响应时间的限制，是因为对（a）来说，1位缓冲满了以后，如果0.0001秒后CPU没有响应，新的数据就把缓冲覆盖了，而对（c）来说，上面的8位寄存器装满以后可以复制到下面，这时候新的数据可以直接覆盖上面的寄存器，只有在上面的数据也写满，也就是0.0008秒后CPU还没有响应，上面的数据才会把下面的给覆盖掉。

缓冲的好处包括(1) 缓和CPU与I/O设备间速度不匹配的矛盾。(2) 减少对CPU的中断频率，放宽对CPU中断响应时间的限制。(3) 提高CPU和I/O设备之间的并行性。题目给的图体现了第（2）条好处。

6 大纲6.磁盘与文件系统

6.1 磁盘调度

先来先服务(FCFS，First Come First Served)：这是一种最简单的磁盘调度算法。它根据进程请求访问磁盘的先后次序进行调度。此算法的优点是公平、简单，且每个进程的请求都能依次地得到处理，不会出现某一进程的请求长期得不到满足的情况。但此算法由于未对寻道进行优化，致使平均寻道时间可能较长。

最短寻道时间优先(SSTF，Shortest Seek Time First)：该算法选择这样的进程：其要求访问的磁道与当前磁头所在的磁道距离最近，以使每次的寻道时间最短。但这种算法不能保证平均寻道时间最短。

扫描(SCAN)算法（电梯调度算法）：当磁头正在自里向外移动时，SCAN算法所考虑的下一个访问对象，应是其欲访问的磁道既在当前磁道之外，又是距离最近的。这样自里向外地访问，直至再无更外的磁道需要访问时，才将磁臂换向为自外向里移动。这时，同样也是每次选择这样的进程来调度，即要访问的磁道在当前位置内距离最近者，这样，磁头又逐步地从外向里移动，直至再无更里面的磁道要访问，从而避免了出现“饥饿”现象。由于在这种算法中磁头移动的规律颇似电梯的运行，因而又常称之为电梯调度算法。

循环扫描(CSCAN)算法：CSCAN 算法规定磁头单向移动，例如，只是自里向外移动，当磁头移到最外的磁道并访问后，磁头立即返回到最里的欲访问的磁道，亦即将最小磁道号紧接着最大磁道号构成循环，进行循环扫描。

(2013.5).（6分）设磁盘的I/O请求队列中的柱面号为：65，68，49，28，100，170，160，48，194.磁头初始位置为110，磁臂方向从小到大，请给出分别采用最短寻道时间优先的磁盘调度算法和电梯磁盘调度算法的柱面移动次数，并给出操作系统采用何种磁盘调度算法更好，为什么？

解答. 最短寻道时间优先：

（从110号开始）	
下一柱面号	移动次数
100	10
68	32
65	3
49	16
48	1
28	20
160	132
170	10
194	24
总移动次数：248	

电梯：

（从110号开始，从小到大）	
下一柱面号	移动次数
160	50
170	10
194	24
100	94
68	32
65	3
49	16
48	1
28	20
总移动次数：250	

采用电梯磁盘调度算法更好，因为它是对最短寻道时间优先算法的改进，可以有效防止老进程出现“饥饿”现象。

注. 做题时可以先画一条数轴，从小到大标好柱面号和初始位置。

最短寻道时间优先算法也不是先往一个方向走到头再掉头往另一个方向走的，比如初始位置110，柱面号28，100，160，500，就是一个左右横跳的过程。

(2014.2).（6分）设磁盘的I/O请求队列中的柱面号分别为：155，158，139，118，190，260，250，138，284，磁头初始位置为200，磁臂方向由小到大。（1）请给出采用SSTF的磁盘调度算法的磁头的柱面移动次数。（2）SSTF的磁盘调度算法有何缺点？

解答. （1）

（初始位置200，由小到大）	
下个柱面号	移动次数
190	10
158	32
155	3
139	16
138	1
118	20
250	132
260	10
284	24
总移动次数：248	

（2）如果不断有新进程的请求到达，且其所要访问的磁道与磁头当前所在磁道的距离较近，这种新进程的I/O请求必然被优先满足，就会让某些老进程出现“饥饿”现象。

注. 这题要求的是最短寻道时间优先算法，因此和磁臂运动方向无关，题干是用来干扰的。

(2015.4(2)). （5分）扫描算法（SCAN）是一种磁盘调度算法，它的优化目标是什么？设磁盘的I/O请求队列中的柱面号依次为：35，58，40，28，80，160，143，38，204，磁头初始位置为95，若采用SCAN（先由小到大开始扫描）磁盘调度算法，磁头移动多少个磁道。

(2018.13(1)(2)). 磁盘请求柱面按10，22，20，2，40，6，38的次序到达，当前磁头在柱面20上。

1）磁盘访问时间由哪几步组成，如何计算？（3分）

2）计算采用SSTF，SCAN算法（先由小到大）磁头移动顺序。（3分）

解答. （1）1.寻道时间：把磁臂移动到指定磁道上所经历的时间。等于启动磁臂的时间与磁头移动到指定磁道所花费的时间之和。2.旋转延迟时间：指定扇区移动到磁头下面所经历的时间。3.传输时间：把数据从磁盘读出或向磁盘写入数据所经历的时间。等于 要读取或写入的字节数/(每秒钟转数*一条磁道上的字节数)

（2）SSTF：

（从20号开始）	
下一柱面号	移动次数
20	0
22	2
10	12
6	4
2	4
38	36
40	2

SCAN：

（从20号开始，从小到大）	
下一柱面号	移动次数
20	0
22	2
38	16
40	2
10	30
6	4
2	4

6.2 廉价磁盘冗余阵列

(2018.13(3)). 如何应用RAID（廉价磁盘冗余阵列）提高磁盘的访问速度，请画图示意。（3分）

解答.

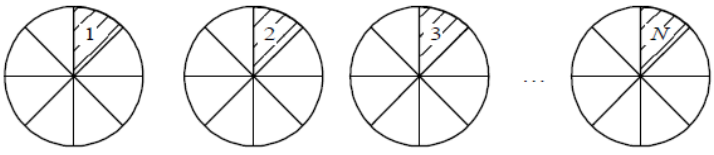


图 5-29　磁盘并行交叉存取方式

如图，在存放一个文件时，可将该文件中的第一个数据子块放在第一个磁盘驱动器上；将文件的第二个数据子块放在第二个磁盘的相同位置上；……；将第N 个数据子块，放在第N个驱动器的相同位置上。以后在读取数据时，采取并行读取方式，即同时从第1～N个数据子块读出数据，这样便把磁盘I/O的速度提高了N-1倍。

6.3 文件系统

(2015.1). （15分）文件系统是操作系统的主要功能之一，请设计一个文件系统，需给出以下信息：

（1）给出描述文件的数据结构（即文件控制块）和目录结构；（2分）

（2）以索引节点为文件系统的物理文件组织结构，图示索引节点结构，说明其优点；（3分）

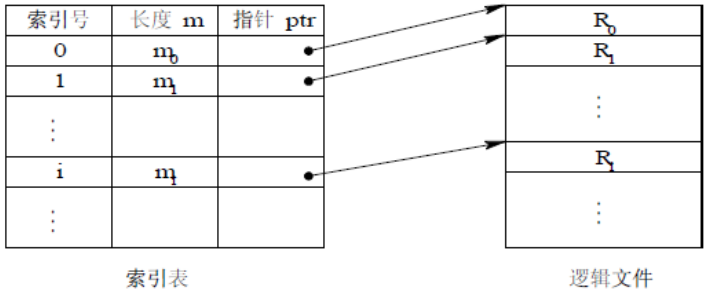
（3）为该文件系统设计几个必要的系统调用，选其中一个为例，详细说明实现该系统调用的方法和过程（注意要使用以上设计中的数据结构）。（10 分）

解答.（1）文件控制块：

文件名	扩展名	物理位置	权限	建立时间	上一次修改的时间

目录结构：单级目录结构：

文件名	物理地址	文件说明	状态位
1			
2			
……			



（2）

优点：有较快的检索速度，主要用于对信息处理的及时性要求较高的场合。

（3）创建文件、删除文件、打开文件、关闭文件。