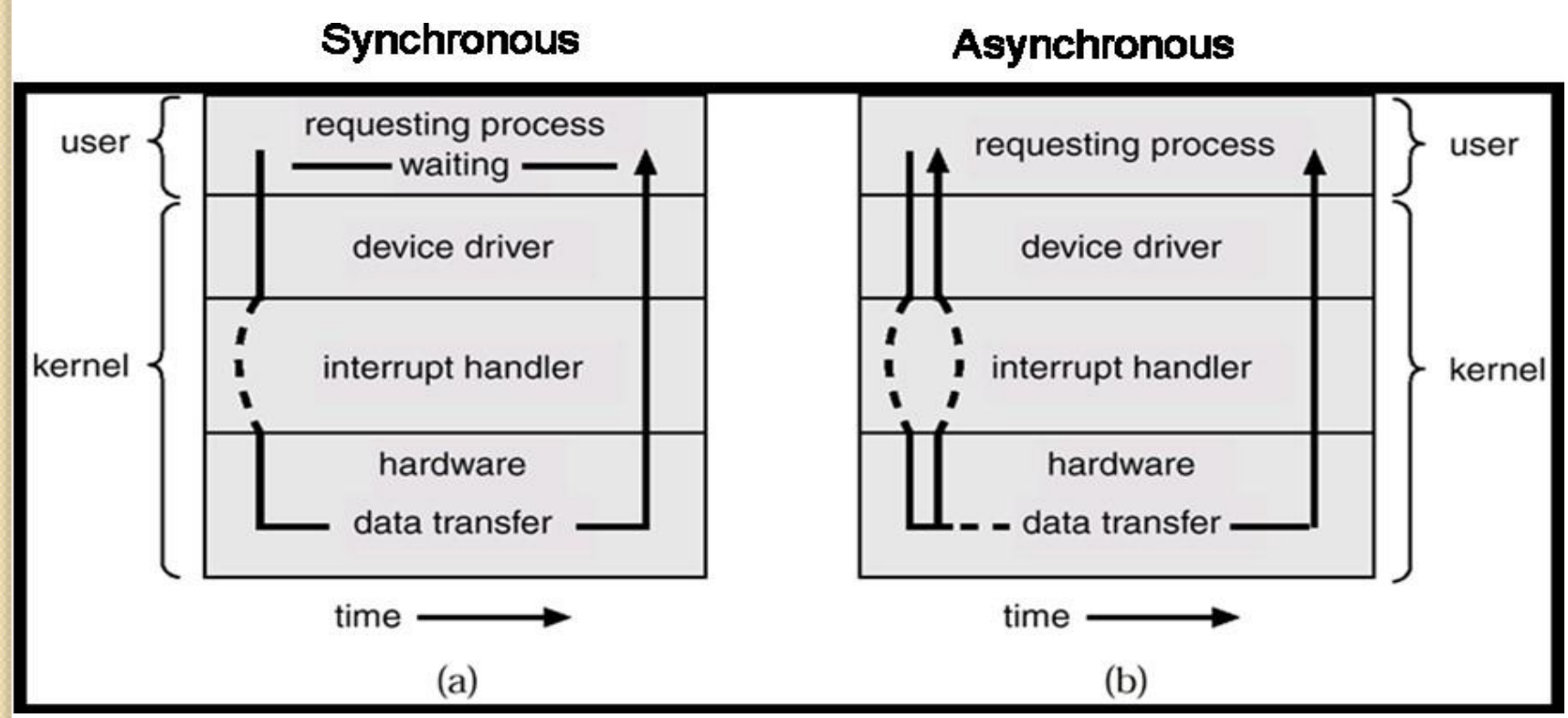
操作系统习题课

By 饶洋辉
raoyangh@mail.sysu.edu.cn

第一章　Introduction



第二章　Structures

主要考察内容：

1）系统调用

2）API

第二章　Structures

通信的两种模式是什么？这两种模式的优点和缺点是什么？

第二章　Structures

通信的两种模式是什么？这两种模式的优点和缺点是什么？

答案：　通信的两种模式是：1）共享内存，2）消息传递。

<消息传递>

优点：　可以用作同步机制来处理进程间的通信；交换的数据量少时，不必避免冲突；也比用于计算机间通信的共享内存更容易实现。

缺点：　消息传递通常包含系统调用。

<共享内存>

优点：　允许通信的最大速度和方便性；进程间通信不需要内核的协助

缺点：　在保护和同步方面仍存在一定问题；没有提供协调通信进程同步的进程。

第三章　Processes

主要考察内容：

1）进程状态变换（五个状态）

2）进程与程序的区别

3）进程调度（长期、中期、短期）

4）进程间协作与通信

第三章　Processes

通常对于批处理系统，进程更多地是被提交，而不是马上执行。这些进程被放到大容量存储设备（通常为磁盘）的缓冲池中，保存在那里以便以后执行。**长期调度程序**（long-term scheduler）或**作业调度程序**（job scheduler）从该池中选择进程，并装入内存以准备执行。**短期调度程序**（short-term scheduler）或 **CPU 调度程序**从准备执行的进程中选择进程，并为之分配 CPU。

有的操作系统如分时系统，可能引入另外的中期调度程序（medium-term scheduler），如图 3.8 所示。中期调度程序的核心思想是能将进程从内存（或从 CPU 竞争）中移出，从而降低多道程序设计的程度。之后，进程能被重新调入内存，并从中断处继续执行。这种方案称为交换（swapping）。通过中期调度程序，进程可换出，并在后来可被换入。为了改善进程组合，或者因内存要求的改变引起了可用内存的过度使用而需要释放内存，就有必要使用交换。交换将在第 8 章讨论。

第三章　Processes

下面设计的优点和缺点分别是什么?

自动和有限缓冲

第三章　Processes

下面设计的优点和缺点分别是什么?

自动和有限缓冲

答案：　自动缓冲：提供了一个无限容量的缓冲，即队列长度可以无限。因此不管多少消息都可以在其中等待，从而保证了发送者在复制消息时不会因为缓冲区已满而遇到阻塞。然而这种方法往往会为了保留足够大的内存而导致资源的浪费。

有限缓冲：队列长度为有限的n，因此最多只能有n个消息驻留其中。若发送消息时队列未满则发送者不必等待不过若队列已满发送者则必须阻塞直到队列中的空间可用为止。但是，这种方法会使得内存被浪费的可能性大大的降低。

第四章　Threads

主要考察内容：

1）线程基本概念

2）多线程模型（多对一、一对一、多对多）

3）用户线程和内核线程的区别

4）为什么要采用线程，线程和进程的区别

第四章　Threads

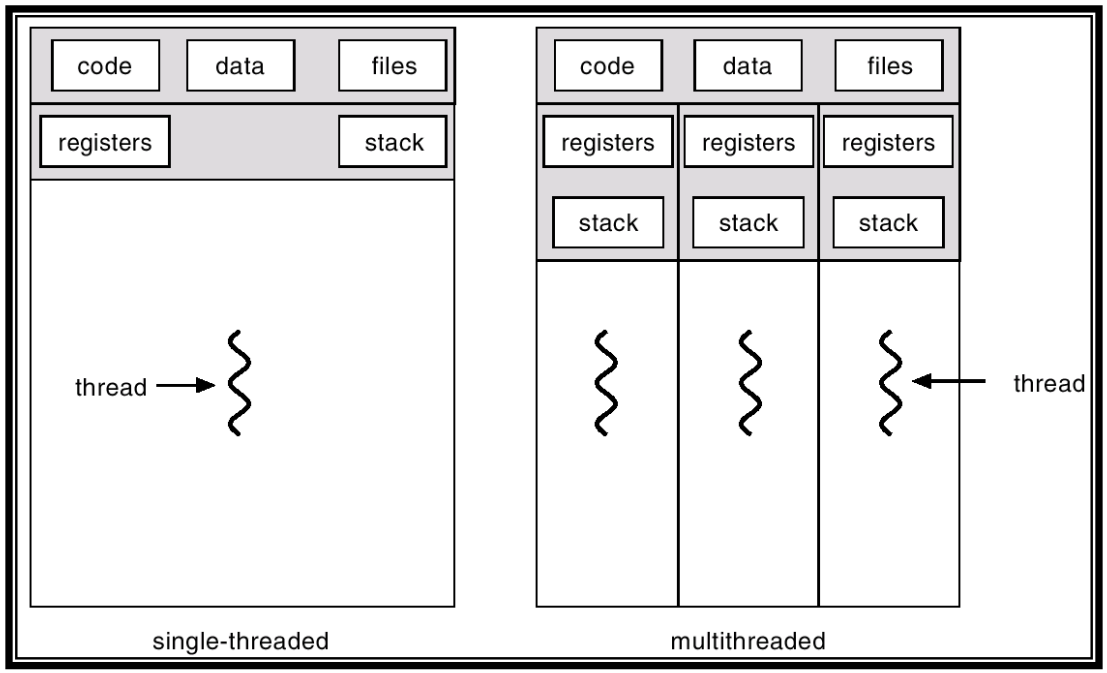
以下进程中的哪些组成部分在多线程程序中是被线程共享的？

a) 寄存器值

b) 堆内存

c) 全局变量

d) 栈内存



第四章　Threads

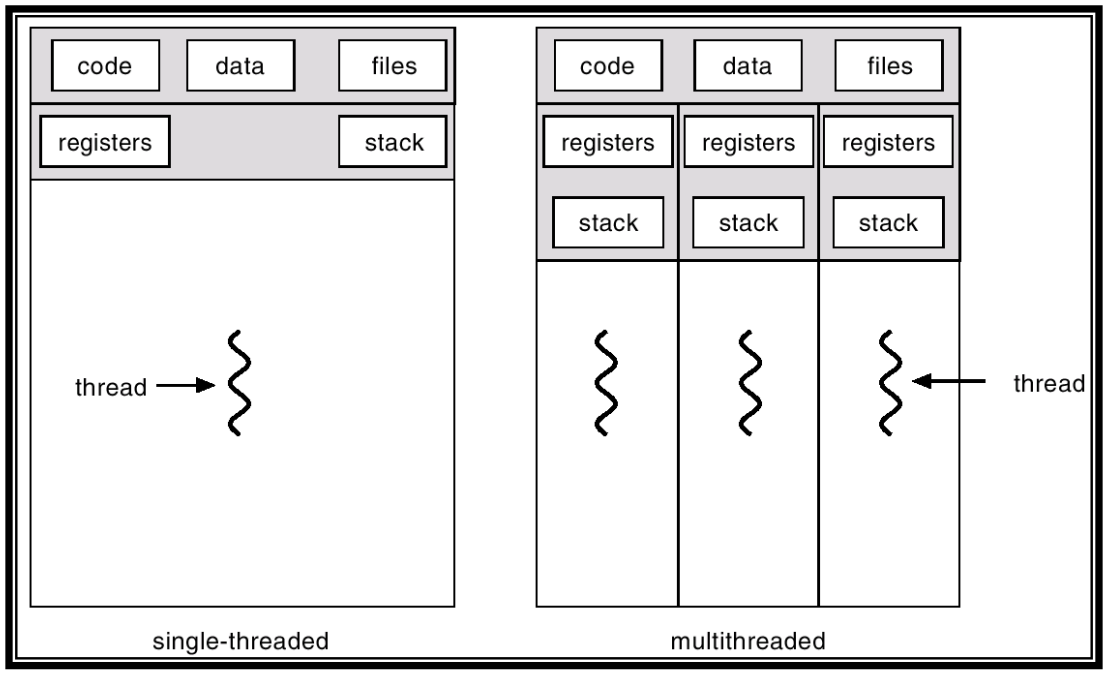
以下进程中的哪些组成部分在多线程程序中是被线程共享的？

a)　寄存器值

b)　堆内存

c)　全局变量

d)　栈内存



答案：　b和c。同一个线程共享堆内存和全局变量，但每个线程都有属于自己的一组寄存值和栈内存。

第四章　Threads

- **State shared by all threads in process/address space**
  - Contents of memory (global variables, heap)
  - I/O state (file system, network connections, etc)

- **State "private" to each thread**, kept in Thread Control Block (TCB)
  - CPU registers (including program counter)
  - Execution stack

第五章　CPU Scheduling

主要考察内容：

1）基本概念（可抢占式等）

2）调度准则（CPU使用率、周转时间、等待时间等）

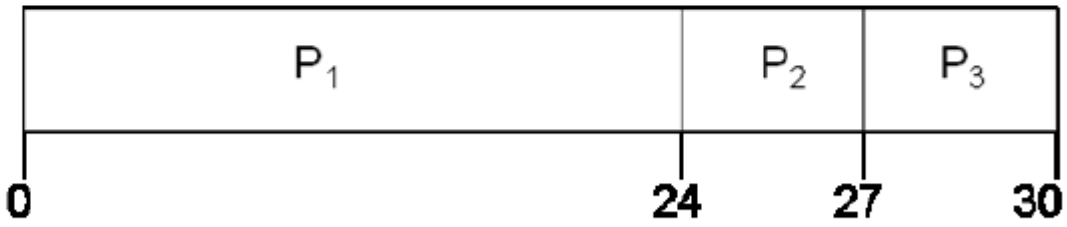
3）调度算法（FCFS、SJF、优先权调度等等）

第五章　CPU Scheduling

- FCFS：最公平的调度思想，谁先来就先服务谁，大家想一想日常生活中的队列，就是这样的一种 FCFS 的典型。

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

第五章　CPU Scheduling

- SJF：最短作业优先，同一批到达的作业，谁的时间短，就先给谁服务，同时分抢占性和非抢占性两种情况。
  - 非抢占性：为您服务，不离不弃。接受到任务后会一直执行到结束。
  - 抢占性："见钱眼开"，如果在执行中如果存在待执行的作业的执行时间短于当前作业的剩余时间，当前作业被抢断。
- RR：时间片轮转。执行人人有份，你方唱罢我登台。注意：如果在一个时间片结束之前，进程已经执行结束，则立即进入下个时间片的调度！（CPU 绝不会干耗完剩余的时间）

第五章　CPU Scheduling

Example of Non-Preemptive SJF

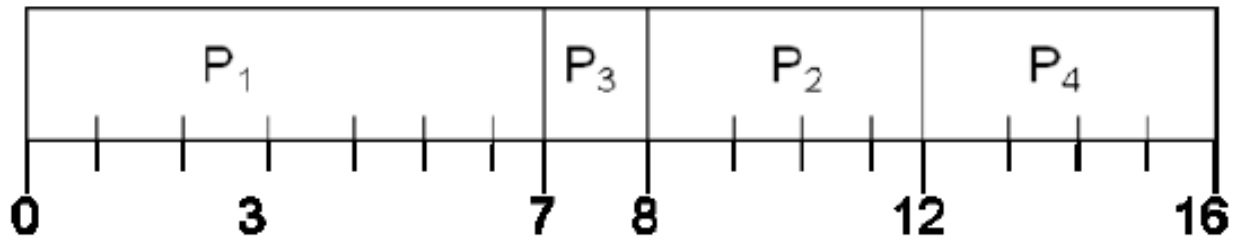
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

第五章　CPU Scheduling

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

第五章　CPU Scheduling

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

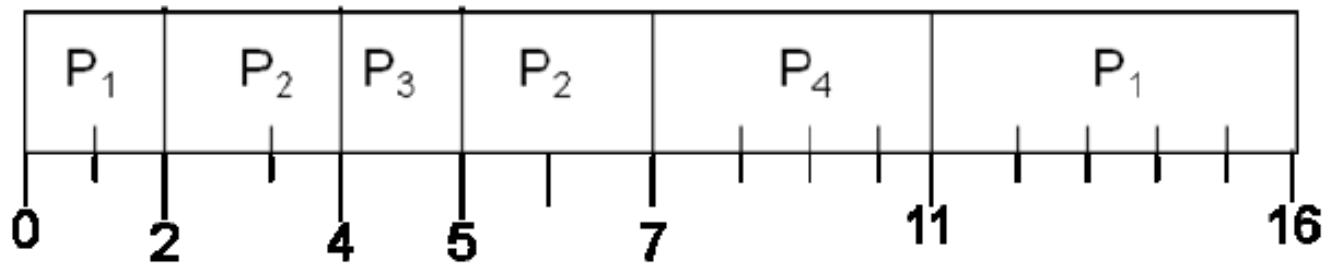
第五章　CPU Scheduling

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

注：P2 被抢断，其等待时间应为 5-4=1；注意，等待时间从某次等待的时刻算起，到再次被执行时结束。

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

第五章　CPU Scheduling

考虑下列进程集，进程占用的CPU区间长度以毫秒来计算：

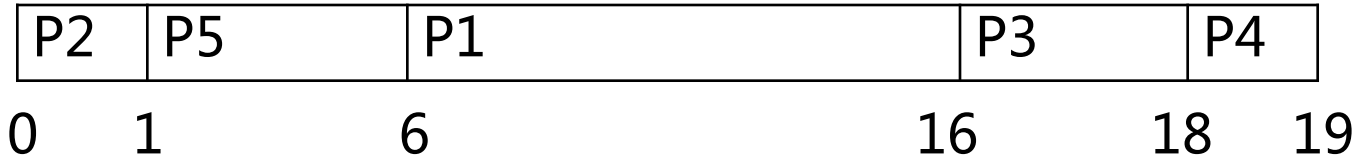
进程	区间时间	优先级
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

假设在时刻0以进程P1，P2，P3，P4，P5的顺序到达。

a.画出4个Gantt图分别演示用FCFS、SJF、非抢占优先级（数字小代表优先级高）和RR（时间片＝1）算法调度时进程的执行过程。

第五章　CPU Scheduling

Priority调度算法（不少同学没按照“同等优先级按照FCFS”的算法来执行）



b.在a里每个进程在每种调度算法下的周转时间是多少？

周转时间：从进程提交到进程完成的时间间隔。

c.在a里每个进程在每种调度算法下的等待时间是多少？

等待时间：在就绪队列中等待所花时间之和。

第六章　Process Synchronization

主要考察内容：

1）基本概念

2）临界区问题的解答（三个条件）

3）信号量

4）经典同步问题（有限缓冲问题、读者-写者问题等）

第六章　Process Synchronization

著名的正确的解决了两个进程的临界区域问题的软件解决方案称为Peterson算法。P0、P1两个进程，共享以下的变量：

boolean flag[2]; /* initially false */

int turn;

进程 Pi(i==0 or1)和另一进程为Pj(j == 1 or 0)的结构见下面代码。证明这个算法满足临界区问题的三个条件。

第六章　Process Synchronization

进程Pi的结构：

```
while(true) {
   flag[i] = true;
   turn = j;
   while(flag[j]&&turn == j){
      忙等待
   }
   临界区
   flag[i] = false;
}
```

进程Pj的结构：

```
while(true) {
   flag[j] = true;
   turn = i;
   while(flag[i]&&turn == i){
      忙等待
   }
   临界区
   flag[j] = false;
}
```

临界区必须满足以下三个条件：互斥、有空让进、有限等待。

（1）互斥：只有当flag[j] = false或者 turn == i时，进程Pi才进入临界区。如果两个进程同时在临界区执行，那么flag[i] ==flag[j] ==true，可是由于turn不可能同时等于0和1，Pi、Pj互斥。

（2）有空让进：若Pi跳出临界区，则必定会执行flag[i] = false，从而使进程Pj跳出循环而进入临界区，从而达到是Pj前进的目的。

（3）有限等待：从（2）可知，Pj最多在Pi进入临界区一次后就可进入，因此Pj的等待次数有限。

第六章　Process Synchronization

忙等待的含义是什么？在操作系统中还有哪些其他形式的等待？忙等待能完全避免吗？给出你的答案。

第六章　Process Synchronization

忙等待是指某个进程循环检查某个条件是否为真，进程不会进入等待状态，并且不主动放弃CPU资源。

好处：不需要上下文切换。**坏处**：占用CPU进行空循环，而这些CPU时间本可以让其他就绪进程使用。

用处：当临界区较短（即较短时间内能得到锁的情况下），忙等待比较有用，否则使用忙等待不高效。

分析：许多同学认为采用了PV信号量，忙等待就能完全避免，其实忙等待并不能完全避免的。只是尽量的减少了。只是将应用程序临界区的忙等限制到wait()和signal()操作的临界区，这些区比较短。因此几乎不占用临界区，忙等很少发生，且所需的时间很短。详细可参考英文版教科书P203最下面一段。

第六章　Process Synchronization

桌上有一个空盘，允许存放一只水果。爸爸可以向盘中放苹果，也可向盘中放桔子，儿子专等吃盘中的桔子，女儿专等吃盘中的苹果。规定当盘空时一次只能放一只水果供吃者使用，请用P、V原语实现爸爸、儿子、女儿三个并发进程的同步。

第六章　Process Synchronization

```
Binary_semaphore
S=1;
Binary_semaphore
Sa=0;
Binary_semaphore
So=0;
Main()
{
begin
    Father()
    Son()
    Daughter();
end;
}
```

```
Father()
{
While(1)
{
P(S);
放水果到盘
中;
If(放的是
桔子)
        V(So))
Else
        V(Sa);
}
}
```

```
son()
{
While(1)
{
P(So);
取桔子
V(S)
吃桔子
}
}
```

```
daughter()
{
While(1)
{
P(Sa);
取苹果
V(S)
吃苹果
}
}
```

第七章　Deadlocks

主要考察内容：

1）死锁四个必要条件

2）死锁预防（破坏某一必要条件）

3）死锁避免（银行家算法）

4）死锁检测算法

第七章　Deadlocks

Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix Need?

b. Is the system in a safe state?

c. If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?

第七章　Deadlocks

a. Need[i] = Max[i] –Allocation[i]

	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

b.是的。系统处于安全状态，因为Available矩阵等于（1 5 2 0），进程P0和P3都可以运行，当进程P3运行完时，它释放它的资源，而允许其它进程运行。

c.可以被满足，满足以后，Available矩阵等于（1 1 0 0），当以次序P0、P2、P3、P1、P4运行时候，可以完成运行。

第七章　Deadlocks

With the banker's algorithm, consider the following snapshot of a system:

resources / Processes	Allocation A B C D	Need A B C D	Available A B C D
P0	0 0 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	0 0 1 4	0 6 5 6	

1. Is the system in a safe state?

2. If a request from process P2 arrives for(1, 2, 2, 2), can the request be granted immediately?

第七章　Deadlocks

（1）利用银行家算法对此时刻的资源分配情况进行分析，可得下表：

	Work				Need				Allocation				Work+Allocation				Finish
	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	1	7	5	0	0	0	1	4	1	9	9	10	True
P1	1	9	9	10	0	6	5	6	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

从上述分析可以看出，此时存在一个安全序列{P0、P3、P4、P1、P2}，故该状态是安全的。

（2）P2提出请求Request2(1,2,2,2)。按银行家算法进行检查：

Request2（1,2,2,2）<=Need（2,3,5,6）

Request2（1,2,2,2）<=Available(1,6,2,2)

试分配并修改相应数据结构后，发现这时可用资源Available（0, 4, 0, 0）已不能满足任何进程的需求，此时系统不能将资源分配给P2。

第七章　Deadlocks

- There is a system with 150 storage units. The storage units were allocated to P1 ~ P3 as the following table at T0. Show how the Banker's Algorithm works to decide whether the system is in a safe state or in an unsafe state.

Process	Max	Allocation
P_1	70	25
P_2	60	40
P_3	60	45

- (1) The 4th process P4 arrived, maximum needs is 60 storage units, current allocation is 25 units.
- (2) The 4th process P4 arrived, maximum needs is 60 storage units, current allocation is 35 units.

第七章　Deadlocks

- （1）安全，回答下列6 种可行的执行中任一序列则可得分
  - 3、1、2、4
  - 3、1、4、2
  - 3、2、1、4
  - 3、2、4、1
  - 3、4、1、2
  - 3、4、2、1

- （2）不安全，因为银行家算法无法满足4 个进程中任一要求，从而产生死锁。

第七章　Deadlocks

- （1）安全，回答下列6 种可行的执行中任一序列则可得分
  - 3、1、2、4
  - 3、1、4、2
  - 3、2、1、4
  - 3、2、4、1
  - 3、4、1、2
  - 3、4、2、1

分析： 许多同学只写出答案，虽然答案正确，但是没有写出分析的过程。注意，在考试中务必写清楚分析的过程以及思路，否则答案错的话只能当全错处理。

- （2）不安全，因为银行家算法无法满足4 个进程中任一要求，从而产生死锁。

第八章　Memory Management

主要考察内容：

1）基本概念（物理地址、逻辑地址、内/外部碎片等）、MMU

2）连续内存分配（首次/最佳/最差适应）

3）分页（逻辑地址映射成物理地址）

4）分段（逻辑地址映射成物理地址）

第八章　Memory Management

按顺序给出5个部分的内存，分别是100KB, 500KB, 200KB, 300KB和600KB，用 first-fit,best-fit和worst-fit算法，能够怎样按顺序分配进程212KB, 417KB, 112KB和426KB？哪个算法充分利用了内存空间？

- **首次适应**：分配*第一个*足够大的孔。查找可以从头开始，也可以从上次首次适应结束时开始。一旦找到足够大的空闲孔，就可以停止。
- **最佳适应**：分配*最小的*足够大的孔。必须查找整个列表，除非列表按大小排序。这种方法可以产生最小剩余孔。
- **最差适应**：分配*最大*的孔。同样，必须查找整个列表，除非列表按大小排序。这种方法可以产生最大剩余孔，该孔可能比最佳适应方法产生的较小剩余孔更为有用。

第八章　Memory Management

由上图知，best-fit算法的内存利用率最高

注意内存被分配后，剩下的内存所产生的洞还是可以被利用的。很多同学都忽略了。

第九章　Virtual Memory

主要考察内容：

1）基本概念

2）页面调度的性能（计算有效访问时间）

3）页面置换算法（FIFO、LRU等）

4）系统颠簸原因以及解决办法

第九章　Virtual Memory

Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming two or three frames? All frames are initially empty.

（1）LRU replacement

（2）FIFO replacement

（3）Optimal replacement

Number of frames	LRU	FIFO	Optimal
2	18	18	15
3	15	16	11

第十一章 File-System Implementation

主要考察内容：

1）基本概念

2）分配方法（连续分配、链接分配、索引分配）

3）空闲空间管理（位向量、链表、组等）

第十二章 Mass-Storage Structure

主要考察内容：

1）磁盘结构

2）磁盘调度算法（FCFS、SSTF、SCAN、C-SCAN、LOOK、C-LOOK）

谢谢！

祝同学们考试顺利