3-7.

semaphore empty=1,full=0;

进程 A_i	进程 B
准备消息	P(full);
P(empty);	从缓冲区 buf 取消息
将消息送入 buf	V(empty);
V(full);	消耗消息

3-8. (IN+1)%100=OUT 时，说明缓冲区满；IN=OUT 时，说明缓冲区空。

3-9. (1) 1 个程序，每个读者一个进程；1 个程序对应 n 个读者进程。

(2) semaphore mutex=1,empty=100;

读者进程：

P(empty);

P(mutex);

在登记表上登记

V(mutex);

进入阅览室阅读

准备离开

P(mutex);

找到自己之前的登记项擦除

V(mutex);

V(empty);

3-14. 答：使用反证法。

假定系统会发生死锁，则根据死锁定义，N 个进程之间会相互阻塞等待，此时 N 个进程至少需要 N 个资源，同时系统已有的 M 个资源也已经分配完毕；因此，此时所有进程的需求总和最少为 N+M（>=N+M），这与题目中“所有进程的需求总和小于 M+N”的表述矛盾，故假设不成立，即在题中所述情况下系统绝不会发生死锁。

3-15. 信号量 S12 表示进程 M1 与 M2 之间的制约关系，初值为 0；同理，设置信号量 S13、S14、S26、S36、S38、S47、S57、S78，初值均为 0。

进程 M1：	进程 M2：	进程 M3：	进程 M4：
……	P(S12);	P(S13);	P(S14);
V(S12);	……	……	……
V(S13);	V(S26);	V(S36);	V(S47);
V(S14);		V(S38);	

进程 M5：	进程 M6：	进程 M7：	进程 M8：
……	P(S26);	P(S47);	P(S38);
V(S57);	P(S36);	P(S57);	P(S78);
	……	……	……
		V(S78)	

3-16. (1) 每个哲学家均只拿到 1 把叉子时将全部吃不上饭。

(2) 详见课件。

3-17.

设置信号量如下：

```
semaphore fmutex=1,          //互斥访问文件
          rdcntmutex=1,      //互斥访问读者计数器
          queue=1,           //互斥访问读写队列
          wmutex=1,          //优先写
          wtcntmutex=1;      //互斥访问写者计数器
int       readcount=0,       //读者计数器
          writecount=0;      //写者计数器
```

读者优先：

```
void reader(){
  while(1){
    P(rdcntmutex);
```

读者与写者公平竞争：

```
void reader(){
  while(1){
    P(queue);
```

写者优先：

```
void reader(){
  while(1){
    P(wmutex);
```

```
    if(readcount==0)
      P(fmutex);
   readcount+=1;
   V(rdcntmutex);
   read operation ...
   P(rdcntmutex);
   readcount-=1;
   if(readcount==0)
     V(fmutex);
   V(rdcntmutex);
  }
}

void writer(){
  while(1){
   P(fmutex);
   write operation ...
   V(fmutex);
  }
}
```

```
    P(rdcntmutex);
    if(readcount==0)
      P(fmutex);
   readcount+=1;
   V(rdcntmutex);
   V(queue)
   read operation ...
   P(rdcntmutex);
   readcount-=1;
   if(readcount==0)
     V(fmutex);
   V(rdcntmutex);
  }
}
void writer(){
  while(1){
    P(queue);
    P(fmutex);
   write operation ...
    V(fmutex);
    V(queue);
  }
}
```

```
    P(rdcntmutex);
    if(readcount==0)
      P(fmutex);
   readcount+=1;
   V(rdcntmutex);
    V(wmutex);
    read operation ...
   P(rdcntmutex);
   readcount-=1;
   if(readcount==0)
     V(fmutex);
   V(rdcntmutex);
  }
}
void writer(){
  while(1){
    P(wrcntmutex);
    if(writecount==0)
      P(wmutex);
    writecount+=1;
    V(wrcntmutex);
    P(fmutex);
   write operation ...
    V(fmutex);
    P(wrcntmutex);
    writecount-=1;
    if(writecount==0)
      V(wmutex);
    V(wrcntmutex);
  }
}
```

3-18.

```
semaphore  customers=0,          //等候理发顾客人数
           barber=0,             //可提供服务理发师人数
           mutex=1;              //互斥访问等候理发顾客计数器
int        waiting=0;            //等候理发顾客计数器
```

理发师进程：

```
while(true) {
    P(customers);
    P(mutex);
    waiting-=1;
    V(mutex);
    V(barber);
    GiveHaircut();
}
```

顾客进程：

```
P(mutex);
if(waiting<n) {
    waiting+=1;
    V(mutex);
    V(customers);
    P(barber);
    WaitingHaircut();
}
else{
  V(mutex);
  exit();
}
```

3-19. 以 n 表示并发进程数，m 表示 n 个并发进程共享 m 个同类资源，x 表示每个进程需要的最大资源数量，则系统处于安全状态时，需满足：n*(x-1)+1<=m。

根据前述条件进行分析可知，(2)和(4)会死锁。

3-20. 更正：第 7 次进程 P3 申请资源数为 3。

(1) 进程 P1 获得 5 个资源后正在运行，进程 P2 获得 2 个资源后阻塞等待分配 2 个资源，进程 P3 阻塞等待分配 4 个资源，系统还剩 3 个资源。

(2) 进程 P1 的请求最先得到满足，运行结束后释放资源，依次唤醒之前的阻塞进程 P3 和 P2；接下来满足进程 P2 的请求后，再唤醒阻塞的进程 P3，最后得到的进程完成序列为：P1、P2、P3。

3-21.

(1) Need 矩阵如下：

Need			
A	***B***	***C***	***D***
0	**0**	**0**	**0**
0	**7**	**5**	**0**
1	**0**	**0**	**2**
0	**0**	**2**	**0**
0	**6**	**4**	**2**

(2) 系统处于安全状态，存在多个安全序列，可得到安全序列包括：(P_0, P_2,-,-,-)，或 (P_2, -,-,-,-)，其中“-”表示可以是任意进程。

(3) 可以满足进程 P2 的申请，此时系统处于安全状态。

注：因笔误，“乐学平台”上的 Available 为(1,5,2,0)，若按此完成第(2)、(3)问，参考答案为：

(2) 系统处于安全状态，存在多个安全序列，可得到安全序列包括：(P_0, P_2,-,-,-)，(P_0, P_3,-,-,-)或 (P_3, -,-,-,-)，其中“-”表示可以是任意进程。

(3) 不可以满足进程 P2 的申请，剩余资源不足以满足进程 P2 的请求。