问答题：
1、 Does Peterson’s solution to the mutual-exclusion problem shown in Fig. 2-24 work when process scheduling is preemptive? How about when it is nonpreemptive?
答：当流程调度是抢占式的时候，Peterson的方法是有效的。
若调度是非抢占式的，那么该方法可能会出现问题。比如turn=0并且先运行P1的时候，循环会一直进行。

2、 In Sec. 2.3.4, a situation with a high-priority process, H, and a low-priority process, L, was described, which led to H looping forever. Does the same problem occur if round-robin scheduling is used instead of priority scheduling? Discuss.
答：使用轮转调度时不会发生这种问题。因为就绪队列中每个进程/线程轮流地运行一个时间片，时间片耗尽时会强迫其让出处理器进行等待，这样低优先级进程必然能够得到运行的机会。然而若是使用优先级调度，L可能根本不会运行。

3、 Consider the following solution to the mutual-exclusion problem involving two processes P0 and P1. Assume that the variable turn is initialized to 0. Process P0’s code is presented below. For process P1, replace 0 by 1 in above code. Determine if the solution meets all the required conditions for a correct mutual-exclusion solution.
答：该方案满足所有要求。因为P0和P1不可能都处在自己的critical section，当turn=0时P0可以执行，但P1不执行，turn=1时同理。但是这种解决方案要求P0先运行，并且P0和P1必须交替运行，这是其缺点所在。

应用题：
课后3：
Case 1：P2的最后一个赋值在P1的最后一个赋值之后进行
x=10 y=9 z=15
Case 2：P1的最后一个赋值在P2的最后一个赋值之后进行
x=10 y=19 z=15
Case 3：P2的后两个赋值在P1的后两个赋值之前进行
x=10 y=9 z=5

课后22：
使用信号量与PV操作：

```
semphore waits,mutex;
int sum=0;
wait=0,mutex=1;

cobegin

process readeri ( var number:integer ; )
begin
P(mutex ) ;
L:if sum+number≥ K then { V ( mutex ) ; P ( waits ) ; goto L ; }
Then sum:sum+number;
V (mutex ) ;
Read file ;
P(mutex ) ;
sum: = sum-number ;
V(waits ) ;
V(mutex ) ;

coend
```

使用管程：

```
TYPE sharefile = MONITOR
int numbersum ,n;
SF : codition ;
DEFINE startread , endread ;

procedure startread (int number) ;
begin check (IM ) ;
L :if（number + numbersum )≥ K then {wait(SF,IM) ; goto L ; }
Numbersum:=numbersum+number;
release (IM ) ;
end

procedure endread (int number) ;
begin
check(IM ) ;
numbersum : = numbersum - number ;
signal ( SF , IM ) ;
release ( IM ) ;
end
begin
numbersum:=0
end .

process-i()
var number : integer ;
begin
number : ＝进程读文件编号；
startread(number);;
read F ;
endread(number) ;
end

cobegin
process-i() ;
coend
```

课后24：
（1）、出于安全状态，因为存在安全序列P0，P3，P4，P1，P2
（2）、不能分配，因为分配后系统处于不安全状态

课后29：每个缓冲区写1次，读n_2次

```
semaphore mutex, empty[n2],full[n2];
int i;
mutex=1;
for(i=0;i<=n2-1;i++){
    empty[i]=m;
    full[i]=0;
}
//以下为需要用到的函数
send(){
    int i;
    for(i=0;i<=n2-1;i++)
```

```
        P(empty[i]);
    P(mutex);
    V(mutex);
    for(i=0;i<=n2-1;i++)
        V(full[i]);
}
Ai(){//所有发送消息的进程类似
    while(1){
        send();
    }
}
receive(i){
    P(full[i]);
    P(mutex);
    V(mutex);
    V(empty[i]);
}
Bi(){//所有接受消息的进程类似
    while(i){
        receive(i);
    }
}

cobegin
A1();
...
An1();
B1();
...
Bn2();
coend;
```