

伪代码题

题

生产者-消费者问题

16. (考研真题) 3 个进程 P_1 、 P_2 、 P_3 互斥地使用一个包含 N ($N > 0$) 个单元的缓冲区。 P_1 每次用 `produce()` 生成一个正整数, 并用 `put()` 将其送入缓冲区的某一空单元中; P_2 每次用 `getodd()` 从该缓冲区中取出一个奇数, 并用 `countodd()` 统计奇数的个数; P_3 每次用 `geteven()` 从该缓冲区中取出一个偶数, 并用 `counteven()` 统计偶数的个数。请用信号量机制实现这 3 个进程的同步与互斥活动, 并说明所定义的信号量的含义。要求用伪代码描述。

```
semaphore empty=N,even=0,odd=0,mutex=1;

P1:
while(1){
    x=produce();
    P(empty);
    P(mutex);
    put(x);
    V(mutex);
    if (x%2==0)
        V(even);
    else
        V(odd);
}

P2:
while(1){
    P(odd);
    P(mutex);
    getodd();
    countodd();
    V(mutex);
    V(empty);
}

P3:
while(1){
    P(even);
    P(mutex);
    geteven();
    counteven();
    V(mutex);
    V(empty);
}
```

17. (考研真题) 某银行提供了 1 个服务窗口和 10 个供顾客等待时使用的座位。顾客到达银行时, 若有空座位, 则到取号机上领取一个号, 等待叫号。取号机每次仅允许一位顾客使用。当营业员空闲时, 通过叫号选取一位顾客, 并为其服务。顾客和营业员的活动过程描述如下。

```
cobegin {  
    process 顾客 {  
        从取号机上获得一个号码;  
        等待叫号;  
        获得服务;  
    }  
    process 营业员 {  
        while (TRUE) {  
            叫号;  
            为顾客服务;  
        }  
    }  
} coend
```

semaphore numget=1, seats=10, custom=0; //numget 是关于取号机互斥的信号量 ; 信号量
seats 是座位的个数 ; 信号量 custom 是顾客的个数

process 顾客 {	process 营业员 {
P(seats); //看有没有空座位	P(custom);
P(numget); //取号	叫号;
取号;	为顾客服务;
V(numget); //取完号后释放取号机	}
V(custom);	
等待叫号;	
V(seats);	
接受服务;	
}	

18. 如图 1-4-1 所示，有 1 个计算进程和 1 个打印进程，它们共享一个单缓冲区，计算进程不断计算出一个整型结果，并将它放入单缓冲区中；打印进程则负责从单缓冲区中取出每个结果并进行打印。请用信号量机制来实现它们的同步关系。

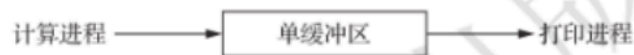


图 1-4-1 共享单缓冲区的计算进程和打印进程

```
semaphore full=0, empty=1;
int buffer;

cp() {
    int nextc;
    while(1) {
        compute the next number nextc;
        P(empty);
        buffer=nextc;
        V(full);
    }
}

pp() {
    int nextp;
    while(1) {
        P(full);
        nextp=buffer;
        V(empty);
        print the number in nextp;
    }
}

main() {
    cobegin
        cp();
        pp();
    coend
}
```

方法总结（重点）

进程同步分析方法（协调进程间的执行顺序）

1. 找出需要同步的代码片段（关键代码）

判断方法：检查哪些操作必须等其他操作完成后才能执行

2. 分析所找代码片段的执行次序

判断方法：明确代码依赖关系，绘制流程图，明确“谁必须等待谁”

3. 增加同步信号量并赋初值

根据依赖关系的数量确定信号量

4. 插入 `wait(S)` 和 `signal(S)` 操作

`wait(S)` 用于检查条件是否满足，`signal(S)` 用于通知条件已满足

进程互斥分析方法（保护共享资源）

1. 查找临界资源

找出被多个进程共享且必须独占访问的资源

2. 划分临界区

确定访问临界资源的具体代码段

3. 定义互斥信号量并赋初值

通常使用 `mutex`（初值=1），表示一次只允许一个进程进入临界区

4. 插入 `wait` 和 `signal` 操作

在临界区前后分别调用 `wait(mutex)` 和 `signal(mutex)`