桌上有空盘，最多允许放一个水果。爸爸可向盘中放一个苹果或一个桔子，儿子专吃桔子，女儿专吃苹果。试用 P、V 操作实现爸爸、儿子、女儿三个并发进程的同步。

```
Semaphore empty = 1, mutex = 1, apple = 0, orange = 0;
Father(){
    while(1){
        P(empty);
        P(mutex);
        将水果放入盘中;
        V(mutex);
        if(放入桔子)
            V(orange);
        else
            V(apple);
    }
}
Son(){
    while(1){
        P(orange);
        P(mutex);
        吃了桔子;
        V(mutex);
        V(empty);
    }
}
Daughter(){
    while(1){
        P(apple);
        P(mutex);
        吃了苹果;
        V(mutex);
        V(empty);
    }
}
void main(){
    Father();
    Son();
    Daughter();
}
```

设某计算进程 CP 和打印进程 IOP 共用一个单缓冲区，CP 进程负责不断地计算数据并送入缓冲区 T 中，IOP 进程负责不断地从缓冲区 T 中取出数据去打印

```
Semaphore data = 0, mutex = 1;
CP(){
    while(1){
        计算数据
        P(mutex);
        CP 进程将数据放入缓冲区 T 中
        V(data);
        V(mutex);
    }
}
IOP(){
    while(1){
        P(mutex);
        IOP 进程将数据取出;
        P(data)
        V(mutex);
        打印数据
    }
}
void main(){
    CP();
    IOP();
}
```

利用管程机制实现读者写者问题

```
monitor Writer_Reader(){
    rcount:int;     //正在读的读者人数
    write:boolean;          //是否有人正在写
    rq,wq:condition;   //申请读和写的队列
    //开始读
    start_read(){
        if (write)
            rq.wait;
        rcount++;
        rq.singal;
    }
    //结束读
    end_read(){
        rcount--;
        if (rcount==0)
            wq.singal;          //没有人读了，则唤醒写者
    }
    //开始写
    start_write(){
        if (rcount > 0 || write)
            wq.write;
        write = true;
    }
    //结束写
    end_write(){
        write = false;
        if (rq != NULL)
            rq.singal;
        else wq.singal;
    }
}

int main(){
    Writer_Reader WR = new Writer_Reader();
    WR.rcount = 0;
    WR.write = false;

    WR.start_read();
    WR.end_read();
    WR.start_write();
    WR.end_write();
}
```