问题

1. 调度?

进程切换是通过*schedule*方法中的*p_proc_ready*切换来实现的，在orange中并没有显示的irq中断分配多核的操作，我们的玩具进程并没有隐式的并行使用时间片

2. milli_delay or delay?

有两个方法*milli_delay*和*delay*方法

```
/* clock.c */
PUBLIC void milli_delay(int milli_sec)
{
        int t = get_ticks();
        while(((get_ticks() - t) * 1000 / HZ) < milli_sec) {}
}
/* klib.c */
PUBLIC void delay(int time)
{
    int i, j, k;
    for(k=0;k<time;k++){
        /*for(i=0;i<10000;i++){ for Virtual PC  */
        for(i=0;i<10;i++){/*    for Bochs   */
            for(j=0;j<10000;j++){}
        }
    }
}
```

注意到get_ticks是共享的，中断切换检查是瞬时的，因而有“**并行**”的结果

3. A-F进程使用的是milli_delay，还是delay，还是系统调用封装函数

皆可，你也可以实现“并发”的*milli_delay*，这些都不作为考察范围，虽然会导致输出不同，但是考察的点主要关注在系统调用封装、读者写者优先实现理解上面，不关注这个，并不会因此加分或扣分

参考：我们实现的是A-E使用milli_delay，F使用系统调用封装

4. 时间片公式?

```
time * 1000 / HZ
```

5. 同时读?

同时读是指我在某个时间片内的控制读写的semaphore资源被使用多少

6. 具体的细节以及方法名字?

自定义，为了**查重**

7. 具体的输出

我们并不会提供输出，读者优先和写者优先已有明显的区别，很多项需求都可以输出并且能够供给检查