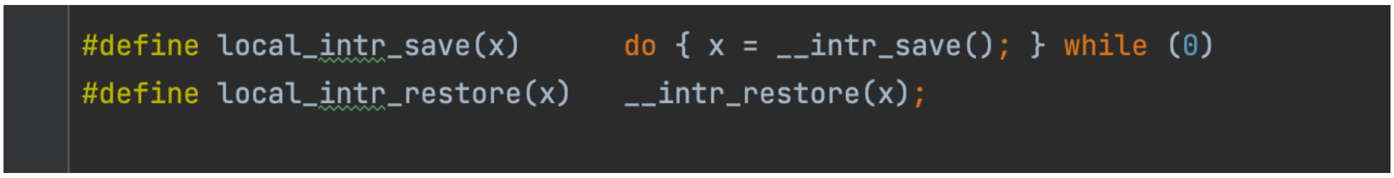
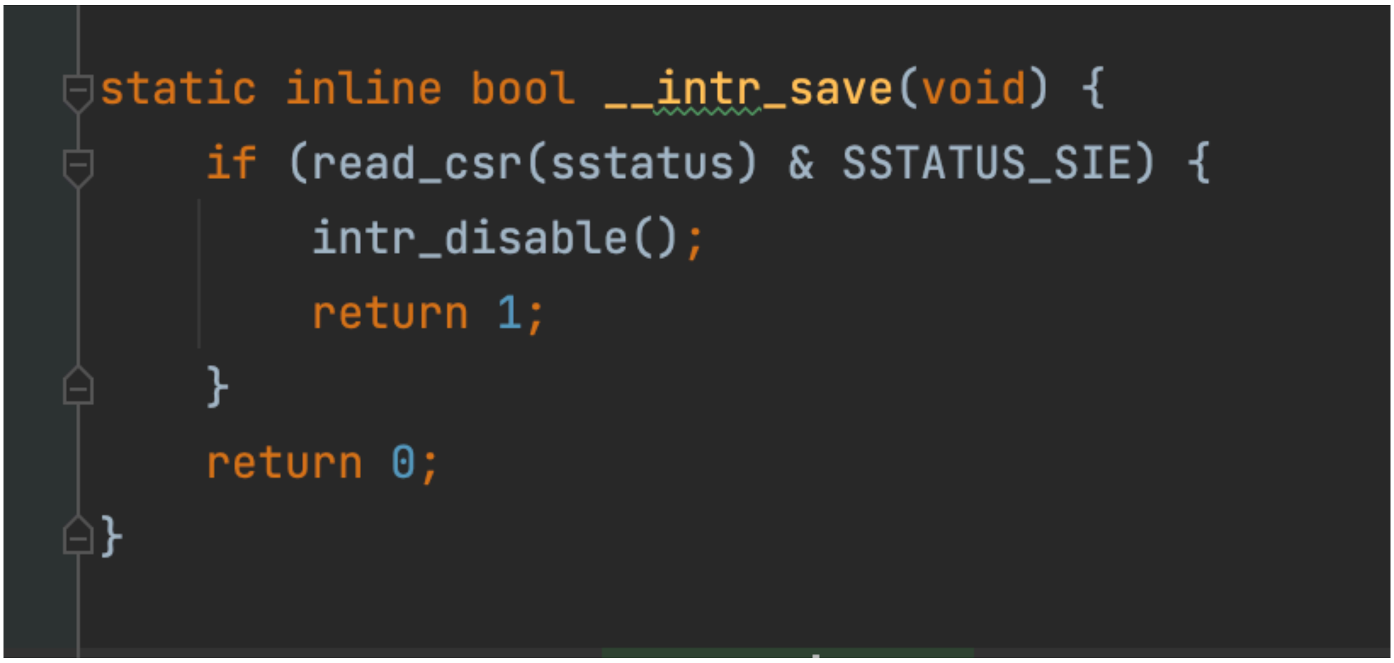
1

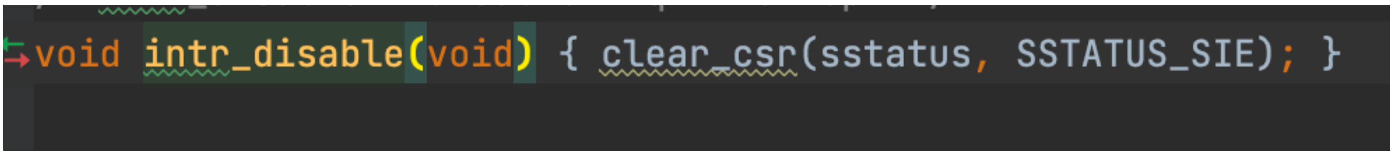


```
#define local_intr_save(x)      do { x = __intr_save(); } while (0)
#define local_intr_restore(x)   __intr_restore(x);
```



```
static inline bool __intr_save(void) {
    if (read_csr(sstatus) & SSTATUS_SIE) {
        intr_disable();
        return 1;
    }
    return 0;
}
```

Local_intre_save() 调用 __intr_save(),如果条件为真，则让使能位激活禁用中断



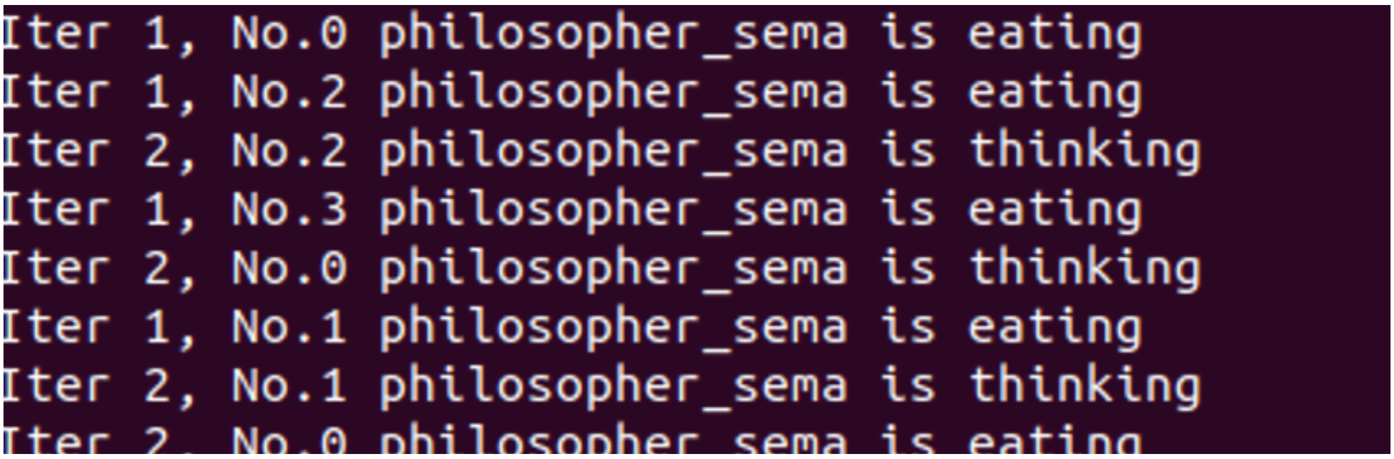
```
void intr_disable(void) { clear_csr(sstatus, SSTATUS_SIE); }
```

2

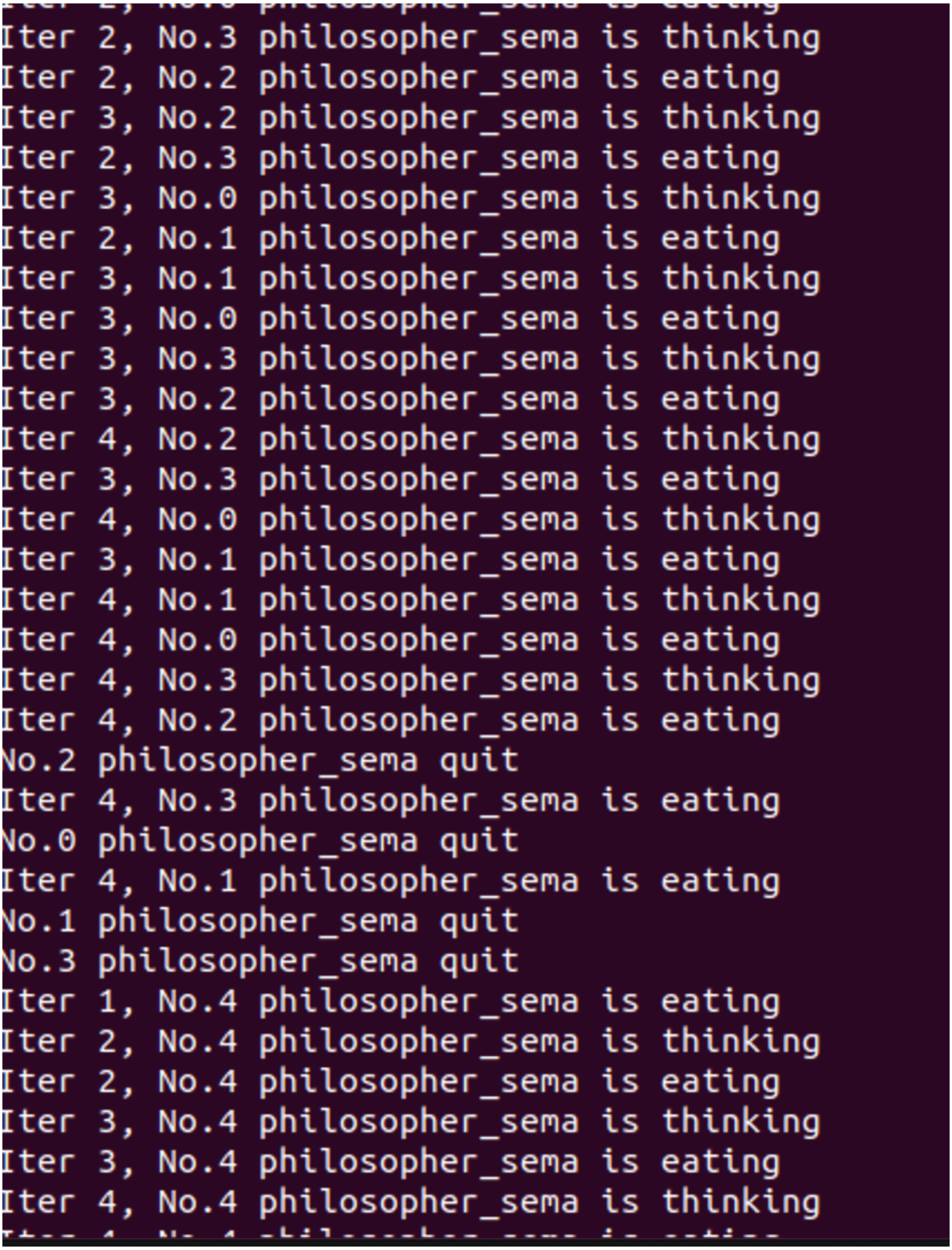
2.1

可以，因为当哲学家拿筷子的时候他会锁住mutex让剩余的哲学家不能动，直到他吃完饭。从而解决死锁的问题。

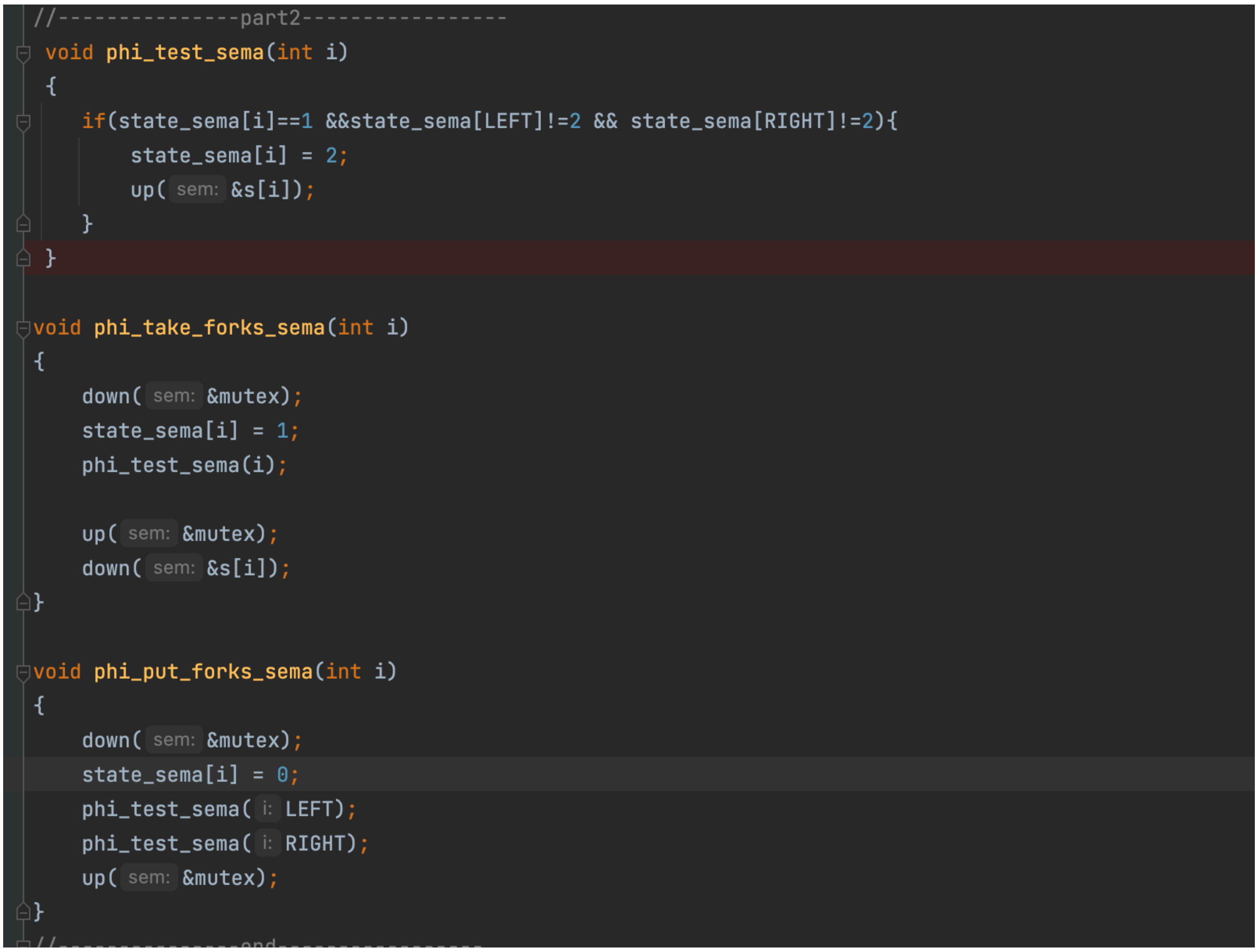
2.2



```
Iter 1, No.0 philosopher_sema is eating
Iter 1, No.2 philosopher_sema is eating
Iter 2, No.2 philosopher_sema is thinking
Iter 1, No.3 philosopher_sema is eating
Iter 2, No.0 philosopher_sema is thinking
Iter 1, No.1 philosopher_sema is eating
Iter 2, No.1 philosopher_sema is thinking
Iter 2, No.0 philosopher_sema is eating
```



```
Iter 2, No.3 philosopher_sema is thinking
Iter 2, No.2 philosopher_sema is eating
Iter 3, No.2 philosopher_sema is thinking
Iter 2, No.3 philosopher_sema is eating
Iter 3, No.0 philosopher_sema is thinking
Iter 2, No.1 philosopher_sema is eating
Iter 3, No.1 philosopher_sema is thinking
Iter 3, No.0 philosopher_sema is eating
Iter 3, No.3 philosopher_sema is thinking
Iter 3, No.2 philosopher_sema is eating
Iter 4, No.2 philosopher_sema is thinking
Iter 3, No.3 philosopher_sema is eating
Iter 4, No.0 philosopher_sema is thinking
Iter 3, No.1 philosopher_sema is eating
Iter 4, No.1 philosopher_sema is thinking
Iter 4, No.0 philosopher_sema is eating
Iter 4, No.3 philosopher_sema is thinking
Iter 4, No.2 philosopher_sema is eating
No.2 philosopher_sema quit
Iter 4, No.3 philosopher_sema is eating
No.0 philosopher_sema quit
Iter 4, No.1 philosopher_sema is eating
No.1 philosopher_sema quit
No.3 philosopher_sema quit
Iter 1, No.4 philosopher_sema is eating
Iter 2, No.4 philosopher_sema is thinking
Iter 2, No.4 philosopher_sema is eating
Iter 3, No.4 philosopher_sema is thinking
Iter 3, No.4 philosopher_sema is eating
Iter 4, No.4 philosopher_sema is thinking
```



```
//---------------part2-----------------
 void phi_test_sema(int i)
 {
     if(state_sema[i]==1 &&state_sema[LEFT]!=2 && state_sema[RIGHT]!=2){
         state_sema[i] = 2;
         up( sem: &s[i]);
     }
 }

void phi_take_forks_sema(int i)
{
    down( sem: &mutex);
    state_sema[i] = 1;
    phi_test_sema(i);

    up( sem: &mutex);
    down( sem: &s[i]);
}

void phi_put_forks_sema(int i)
{
    down( sem: &mutex);
    state_sema[i] = 0;
    phi_test_sema( i: LEFT);
    phi_test_sema( i: RIGHT);
    up( sem: &mutex);
}
//---------------end------------------
```

当拿起叉查看周围是否需要，如果需要就等待、放下时通知周围人