## Esercizi su OS161 (tratti da compiti di esame)

1. Si spieghi, in relazione alla funzione `syscall()`, che cosa rappresenta la sua variabile `callno`, a cui si assegna il valore `tf->tf_v0`. Si dica poi che cosa significano le seguenti istruzioni alla fine della funzione `syscall()` (si dica, in particolare, cosa sono i campi `tf_v0` e `tf_a3` del trapframe):

```
if (err) {
     tf->tf_v0= err;
     tf->tf_a3= 1;
}
else {
     tf->tf_v0= retval;
     tf->tf_a3= 0;
}
```

2. Si consideri la realizzazione dei `lock` in un sistema OS161. Quale thread deve essere considerato owner (proprietario) di un lock? Motivare la scelta.

- Il thread che ha creato il lock.
- L'ultimo thread chiamante la funzione `lock_acquire`.
- Altro (completare)

3. Sono date le funzioni `lock_release` e `lock_do_i_hold` proposte in figura. Nelle funzioni potrebbero essere presenti errori: in caso affermativo, li si identifichi e li si corregga, motivando.
N.B.: si considerino solamente errori logici/funzionali, non di mistyping o sintassi.

```
void lock_release(struct lock *lock) {
     KASSERT(lock != NULL);
     spinlock_acquire(&lock->lk_lock);
     KASSERT(lock_do_i_hold(lock));
     lock->lk_owner=NULL;
     wchan_wakeone(lock->lk_wchan, &lock->lk_lock);
     spinlock_release(&lock->lk_lock);
}
```

```
bool lock_do_i_hold(struct lock *lock) {
     spinlock_acquire(&lock->lk_lock);
     if (lock->lk_owner==curthread)
          return true;
     spinlock_release(&lock->lk_lock);
     return false;
}
```

4. Dato il codice delle funzioni di semaforo P e V riportate di seguito:

```
void P(struct semaphore *sem) {
  spinlock_acquire(&sem->sem_lock);
  while (sem->sem_count == 0) {
     wchan_sleep(sem->sem_wchan,
                 &sem->sem_lock);
  }
  sem->sem_count--;
  spinlock_release(&sem->sem_lock);
}
```

```
void V(struct semaphore *sem) {
  spinlock_acquire(&sem->sem_lock);
  sem->sem_count++;
  KASSERT(sem->sem_count > 0);
  wchan_wakeone(sem->sem_wchan,
                &sem->sem_lock);
  spinlock_release(&sem->sem_lock);
}
```

Si risponda alle seguenti domande:

- A cosa serve lo `spinlock` (in entrambe le funzioni)?
- Perchè la P contiene un ciclo `while` invece di un `if(sem->sem_count == 0)`?
- Perchè il ciclo non è presente della V?
- Perché la `wchan_sleep` riceve come parametro lo `spinlock`? Vale lo stesso motivo per la `wchan_wakeone`?
- È possibile che la chiamata alla `wchan_wakeone` svegli più di un thread in attesa su `wchan_sleep`? Se NO, perché? Se SI, come si fa in modo di rilasciare un solo thread in attesa su P?