

Implementazione lock

Uniprocessore

Per garantire mutua esclusione è sufficiente disabilitare gli interrupt (impedendo quindi il context switch), ma:

- tenerli disabilitati riduce la reattività del sistema, e
- codice utente buggato o malevolo può monopolizzare il processore.

```

acquire() {
    disable_interrupts();
    if (value == BUSY) {
        waiting.add(TCB);
        suspend();
    } else {
        value = BUSY;
    }
    enable_interrupts();
}

release() {
    disable_interrupts();
    if (!waiting.empty()) {
        t = waiting.remove();
        sched.add_ready(t);
    } else {
        value = FREE;
    }
    enable_interrupts();
}

```

suspend: context switch, rimozione del thread dai ready, abilitazione interrupt e rilascio spinlock (sotto).

Multiprocessore

Spinlock con istruzioni RMW:

```
spinlock_acquire() {                                spinlock_release() {
    while (test_and_set(spinlock));                  spinlock = 0;
}                                                    WMB();
}
```

usato per proteggere la sezione critica di acquire e release:

```

acquire(){
    spinlock_acquire();
    disable_interrupts();
    if (value == BUSY){
        waiting.add(TCB);
        suspend();
    } else {
        value = BUSY;
        spinlock_release();
        enable_interrupts();
    }
}

release() {
    spinlock_acquire();
    disable_interrupts();
    if (!waiting.empty()) {
        t = waiting.remove();
        sched.add_ready(t);
    } else {
        value = FREE;
    }
    spinlock_release();
    enable_interrupts();
}

```