10/06/2025 - Sistemas Operativos - Ude@

1. Pendientes:

a. Parcial 1. ✓ DoF = Nota + (0 - 0.7)
Bonus

b. Actividades bonus
- Simulacion 1 (0.2)
- Simulacion 2 (0.2)
- Resumen (0.2) + [0.1] votacion

c. Bonus curso RHA (Ude@ - Bonus) - Pendiente link diplomas

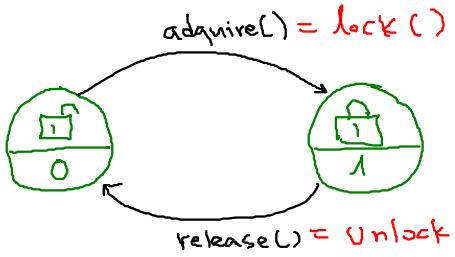
d. Parcial 2 [M1 | M2 | M3 | M4]

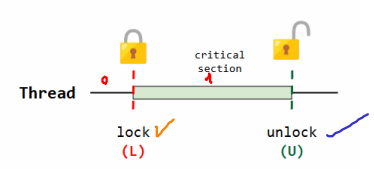
e. Proyecto Final P1 - P2 - P3 - seguimiento Proyecto

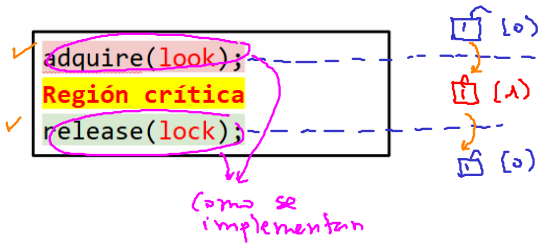
f. Quizes semanales
- semana 8 ✓
- semana 9 ✓
- semana 10 ✓
- semana 11?

2. Implementacion de Locks

Aseguran exclusion Mutua







Implementación de locks

¿Cómo construir un lock?

- ¿Cómo construir un lock que sea eficiente?
- ¿Qué hardware es necesario (Primitivas del ISA, su uso, etc)?
- ¿Cuál es el soporte del SO (Qué necesitamos usar de este)?

Objetivo: Obtener exclusión mutua con el menor costo posible.

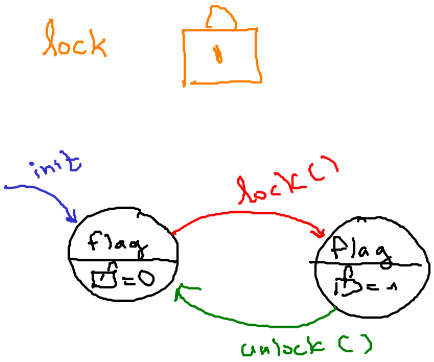
		semaphores	
SW	locks	monitors	Condition variables
HW	loads	stores	test&set
		Disable interrupts	...

```
void lock() {
  // Instrucciones lock
  // ...
}

void unlock() {
  // Instrucciones lock
  // ...
}
```

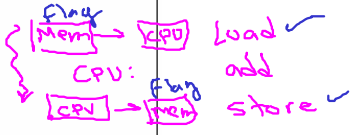
① Uso de interrupciones

```
void lock() {
  DisableInterrupts();
}

void unlock() {
  EnableInterrupts();
}
```

② Uso de load – stores ✗




```
1  typedef struct __lock_t {
2    int flag;
3  } lock_t;
4
5  void init(lock_t *mutex) {
6    // 0 -> lock is available, 1 -> held
7    mutex->flag = 0;
8  }
9
10 void lock(lock_t *mutex) {
11   while (mutex->flag == 1) // TEST the flag
12     ; // spin-wait (do nothing)
13   mutex->flag = 1; // now SET it!
14 }
15
16 void unlock(lock_t *mutex) {
17   mutex->flag = 0;
18 }
```

- No sirve para que la exclusión mutua no se satisfase
- Costosa: spin wait

③ Uso de instrucciones Atomicas (Hw)

a. Test And Set ✓
b. Compare And Swap ✓✓
c. LL & CS ✓✓
