

Exercise walk-through

7.3

Synchronisation

University Computer Science

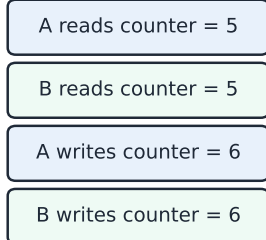
Synchronisation: the lost update

- 1 **Race** Both threads read 5, both write 6 - one increment lost; with four each, every final value 1 to 8 is reachable.
- 2 **Rules Mutual exclusion**, progress, bounded waiting; Peterson and test-and-set both **spin** - mutex queues sleep instead.
- 3 **Deadlock** Mutual exclusion + hold-and-wait + no preemption + circular wait: break one, and a fixed **lock** order is the cheapest.

Race interleaving and lock

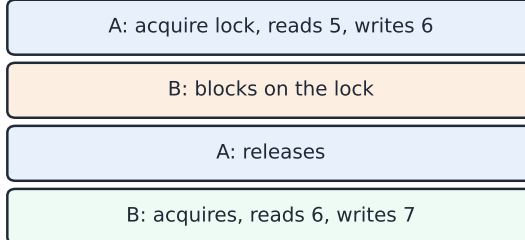
read-modify-write, with and without the lock

unlocked — a lost update



final counter: 6, not 7

locked — the second thread waits



final counter: 7, always

with four increments each, every final value from 1 to 8 is reachable unlocked — the lock pins it at 8

deadlock needs: mutual exclusion + hold-and-wait + no preemption + circular wait — break one

Unlocked: both read 5, both write 6. Locked: the second **thread** waits.

Exercises 2.1-2.3

2.1 Two threads run $x = x + 1$ on $x = 10$. Give the outcome of the interleaving R1 R2 W2 W1. [3]

2.2 Name the three properties a correct critical-section solution must satisfy. [3]

2.3 Why does a mutex queue beat a spinlock on a single core? [3]

Exercises 2.1-2.3

2.1 Two threads run $x = x + 1$ on $x = 10$. Give the outcome of the interleaving R1 R2 W2 W1. [3]

2.1

- Both reads see 10; the last write is 11 – final 11 ✓.

2.2 Name the three properties a correct critical-section solution must satisfy. [3]

2.2

- Mutual exclusion, progress, bounded waiting ✓.

2.3 Why does a mutex queue beat a spinlock on a single core? [3]

2.3

- The waiter sleeps in the OS queue instead of spinning – a spinning thread on the only core blocks the lock holder itself ✓.

Exercise 3.1

3.1 Two threads each execute `count = count + 1` three times on `count = 0`, with per-increment registers.

State the maximum final value and the interleaving that achieves it. State the minimum final value and explain the interleaving that achieves it. Give the number of distinct final values reachable, and the one-line fix that pins the result. [8]

Exercise 3.1

3.1 Two threads each execute `count = count + 1` three times on `count = 0`, with per-increment registers.

State the maximum final value and the interleaving that achieves it. State the minimum final value and explain the interleaving that achieves it. Give the number of distinct final values reachable, and the one-line fix that pins the result. [8]

3.1

- Maximum 6: all six increments land in order, each read sees the prior write ✓.
- Minimum 1: all six reads happen before any write (all read 0), then the six writes each store 1 ✓.
- Distinct final values: 6 – every value 1 to 6 is reachable ✓.
- Fix: wrap the increment in a lock (mutual exclusion) – the result is always 6 ✓.

Exercise 3.2

3.2 Thread A locks L1 then wants L2; thread B locks L2 then wants L1. Each lock acquisition is 1 ms; each hold lasts 5 ms.

Explain why this deadlocks, naming the condition cycle. State the one-line ordering rule that prevents it, and trace the same run under the rule. Give one other condition that could be broken instead, with its mechanism. [7]

Exercise 3.2

3.2 Thread A locks L1 then wants L2; thread B locks L2 then wants L1. Each lock acquisition is 1 ms; each hold lasts 5 ms.

Explain why this deadlocks, naming the condition cycle. State the one-line ordering rule that prevents it, and trace the same run under the rule. Give one other condition that could be broken instead, with its mechanism. [7]

3.2

- A holds L1, waits for L2; B holds L2, waits for L1: each waits forever for the other – circular wait across two holders ✓.
- Rule: always acquire locks in one global order (L1 before L2) ✓.
- Under the rule: B, holding nothing yet, takes L1 first – both threads queue on L1 and proceed one at a time ✓.
- Alternative: break hold-and-wait – request all locks at once (acquire-or-release) so no thread holds while waiting ✓.